CORE
Metadata, citation and similar papers at core.ac.uk
Provided by DSpace at VSB Technical University of Ostrava

Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

DIPLOMOVÁ PRÁCE

2010 David Sedlík

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Návrh a implementace systému pro konstrukci modelu byznys procesu

Design and Implementation of System for Constructing of Business Process Model

2010 David Sedlík

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2010 .....

David Sedlík

Mé poděkování za poskytnuté materiály a odborné rady při zpracování práce patří mému vedoucímu diplomové práce Ing. Davidovi Ježkovi, Ph.D.

Abstrakt

Diplomová práce popisuje návrh a implementaci aplikace pro vytváření modelů podnikových procesů na základě kompozice základních stavebních bloků nebo již existujících procesů a vysvětluje použití matematického modelu Chu space pro výpočet samotné kompozice. Aplikace je postavena na platformě Eclipse a sestavena jako RCP (Rich Client Platform) aplikace. Pro syntaktickou analýzu regulárního výrazu, který definuje podnikový proces, je použit nástroj ANTLR (ANother Tool for Language Recognition). Výsledný diagram zobrazující jednotlivé procesy je sestaven jako vícedimenzionální automat.

Klíčová slova

Chu space, vícedimenzionální automat, podnikový proces, Eclipse Rich Client Platform.

Abstract

This diploma thesis describes the design and implementation of an application for creating business process models based either on the composition of basic construction blocks or existing processes. The business process defined by a regular expression is parsed using the ANTLR (ANother Tool for Language Recognition) tool and the final composition is calculated by the Chu space mathematical model, which is also explained. This RCP (Rich Client Platform) application is built on the Eclipse platform with the resulting diagram showing individual processes as multidimensional automata.

Keywords

Chu space, multidimensional automata, business process, Eclipse Rich Client Platform.

Seznam použitých symbolů a zkratek

RCP	Rich Client Platform
ANTLR	ANother Tool for Language Recognition
EMF	Eclipse Modeling Framework
GEF	Graphical Editing Framework
XML	Extensible Markup Language
IDE	Integrated Development Environment
OSGi	Open Services Gateway initiative
UML	Unified Modeling Language
BPD	Business Process Designer
XMI	XML Metadata Interchange
MVC	Model-View-Cotroller
JAR	Java Archive
HTML	HyperText Markup Language
BMP	Microsoft Windows Bitmap
JPEG	Joint Photographic Experts Group
PNG	Portable Network Graphics

Obsah

Seznam obrázků

Seznam příloh

Seznam obrázků v přílohách

1 Úvod

Podnikové procesy jsou v dnešní době nedílnou součástí každé rozvíjející se firmy či společnosti. Firma má na základě těchto jasně definovaných procesů stanoveny postupy, kterými se ve své podnikatelské činnosti řídí. Tyto procesy jsou poté dynamicky se vyvíjející firmou dále rozšiřovány, zdokonalovány anebo jinak upravovány a to pro dosažení zlepšení ať už kvality, rychlosti, schopnosti adaptovat se na změny atd.

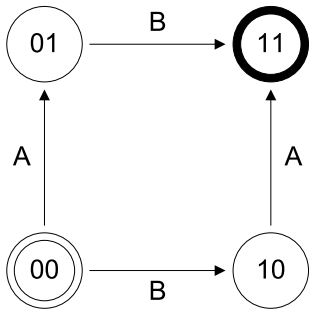
Za účelem modelování podnikových procesů již existují jazyky, kterými jsme schopni procesy popisovat. Jedná se například o jazyk UML (Unified Modeling Language), Petriho sítě a konečné automaty. Zde můžeme modelovat přechody mezi aktivitami, událostmi nebo stavy. V žádném z těchto přístupů však nejsme schopni zachytit obecný souběh aktivit jednoznačnými stavy (částmi diagramu), kde by bylo jasně dáno, které aktivity ještě nebyly spuštěny, které právě probíhají a které již doběhly. Tuto možnost přináší použití vícedimenzionálních automatů, jež bude popsáno v kapitole 2.

K reprezentaci modelu vícedimenzionálního automatu bude využito matematické teorie Chu space popsané v kapitole 3. Chu space poskytuje řadu operací využitelných v procesní algebře. Těmito operacemi lze reprezentovat kompozice procesů. Každý proces skládající se z aktivit, tak i samotnou aktivitu realizovanou pomocí Chu space, lze dále skládat s dalšími takovými Chu space.

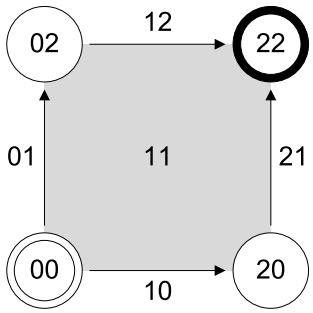
Cílem práce je implementovat aplikaci, která poskytne možnost definování podnikového procesu reprezentovaným jako Chu space a základních operací procesní algebry. Takto definovaný proces zadaný regulárním výrazem bude dále možno použít následovně. Pro vytváření nových procesů. Pro zobrazení jeho grafické podoby pomocí vícedimenzionálního automatu. A pro zobrazení jeho matice Chu space (obsahující všechny dosažitelné stavy automatu), jelikož práce řeší grafickou reprezentaci procesu pouze pomocí bodů a hran, nejsou v grafické podobě zobrazeny všechny stavy automatu. Řešením grafického zobrazení objektů *n*-dimenzionálního prostoru pro potřeby modelování podnikových procesů by se mohla zabývat další práce navazující na níže zmíněnou teorii.

2 Vícedimenzionální automaty

Zásadním přínosem vícedimenzionálních automatů v procesní algebře je možnost zachycení paralelizmu aktivit, jak je v (1) odkazováno na (2). Klasický automat není schopen problematiku paralelního zpracování dvou či více aktivit zachytit. Lze jím pouze znázornit zpracování jednotlivých aktivit za sebou v různém pořadí. Jak je vidět na obr. 2.1, paralelní zpracování je znázorněno tak, že se nejprve provede aktivita *A* a poté aktivita *B* anebo naopak. Stavy konečného automatu jsou vlastně body představující *0*-dimenzionální objekty a přechody mezi nimi *1*-dimenzionální objekty hran. Oproti tomu jsou vícedimenzionální automaty rozšířeny o objekty v *n*-dimenzionálním prostoru. Souběh dvou aktivit je tedy realizován *2*-dimenzionálním objektem, plochou, jak je znázorněno na obr. 2.2. Každý *n*-dimenzionální prvek automatu představuje stav, ve kterém se automat nachází, čili jednoznačně určuje, stav všech aktivit, jejichž průběh modeluje. V konkrétním příkladu souběhu dvou aktivit je to dáno dvojící symbolů (pro každou aktivitu jeden) v každém stavu automatu. Symboly jsou brány z množiny $\{0, 1, 2\}$, kde 0 značí, že aktivita ještě nezačala, 1 znamená právě probíhající aktivitu a 2 aktivitu, která již doběhla.



Obr. 2.1: Souběh aktivit realizovaný klasickým automatem



Obr. 2.2: Souběh aktivit realizovaný vícedimenzionálním automatem

Obecně můžeme říci, že zobrazení n souběžných aktivit vícedimenzionálním automatem je dosaženo v *n*-dimenzionálním prostoru. Definice 1 byla citována v (1), je převzata z (3) a formálně definuje vícedimenzionální automat.

Definice 1. Vícedimenzionální automat je sedmice $(S, d, \sigma, \tau, s_0, F, \ell)$, kde

- S je množina stavů
- $d: S \to \mathbb{N}$ určuje dimenzi stavu
- $\sigma, \tau: S \times \mathbb{N} \to S$ jsou částečné funkce. Pro $s \in S$ a $k < d(s)$, $\sigma(s,k)$ a $\tau(s,k)$ jsou počáteční a koncové stavy akce v *k*-té dimenzi. Funkce σ, τ musí splňovat *kubická* pravidla (4):

  Pro $i \leq j$

  $d\big(\sigma(s,k)\big) = d\big(\tau(s,k)\big) = d(s) - 1$

  $\sigma(\sigma(s,i),j) = \sigma(\sigma(s,j+1),i)$

  $\sigma(\tau(s,i),j) = \tau(\sigma(s,j+1),i)$

  $\tau(\sigma(s,i),j) = \sigma(\tau(s,j+1),i)$

  $\tau(\tau(s,i),j) = \tau(\tau(s,j+1),i)$
- $s_0 \in S$ je počáteční stav a $F \subseteq S$ je množina koncových stavů. Musí platit $(s_0) = d(f_i) = 0$ pro všechna $f_i \in F$.
- $\ell: S \longrightarrow \Sigma$ je značkovací funkce, která přiřazuje popis všem stavům dimenze 1. To znamená, že $\ell(s)$ je definována pouze tehdy, jestliže $d(s) = 1$. Navíc je vyžadováno, aby $\ell\big(\sigma(s,i)\big) = \ell\big(\tau(s,i)\big)$ pro $i = 0{,}1$

Z této definice vícedimenzionálního automatu také vyplývá, že všechny jeho části jsou chápány jako stavy, tedy i hrany, plochy atd., čímž se vícedimenzionální automaty liší od předchozího pojetí konečných automatů.

3 Chu space

V této kapitole se budeme věnovat základní teorii Chu space, která bude využita v implementaci aplikace. V (5) je tato teorie detailně popsána, včetně toho, jak pomocí Chu space reprezentovat vícedimenzionální automat. Díky tomu můžeme operace nad Chu space využít pro procesy vyjádřené vícedimenzionálním automatem. Chu space je jednoduchá matice nad množinou Σ, a to je dvourozměrné pole, jehož hodnoty jsou brány z Σ. Definice 2 je formální definice Chu space citována z (5).

Definice 2. Chu space $\boldsymbol{A} = (A, r, X)$ nad množinou Σ, nazývanou abeceda, se skládá z množiny bodů A, které tvoří nosič (carrier), a množiny X z Σ, která tvoří spolunosič (cocarrier), a funkce $r: A \times X \to \Sigma$ definující matici.

Abeceda může být i prázdná nebo jednoprvková. Počínaje abecedou $\Sigma = \{0, 1\}$ lze pomocí Chu space reprezentovat širokou škálu strukturovaných matematických objektů včetně následujících:

- relační struktury jako množiny a orientované grafy,
- algebraické struktury jako grupy, okruhy, tělesa, vektorové prostory a Booleovy algebry,
- topologie jako například topologické prostory, Hausdorffovy prostory a topologické Booleovské algebry.

V této práci využijeme Chu space nad abecedou $\{0, 1, 2\}$ pro reprezentaci modelu vícedimenzionálního automatu, což je vysvětleno v kapitole 3.2.

3.1 Operace s Chu space

V této kapitole budou popsány čtyři základní operace procesní algebry, které lze realizovat pomocí Chu space. Tyto operace byly vybrány k implementaci v realizované aplikaci.

Nad Chu space je definováno mnoho operací. Tyto operace vycházejí z lineární logiky (6) a procesní algebry zpracované nad Chu space v (7) a (8).

3.1.1 Tensorový součin

Tensorový součin neboli tensor (definice 3) je operace vzešlá z lineární logiky a v procesní algebře může být chápán jako „řádný výskyt“ (orthocurrence) (9). Tuto kombinaci procesů lze označit jako „průtok skrz“ (flow through) jednoho procesu druhým. Například po sobě jedoucí vlaky, které chtějí projet systémem stanic anebo digitální signály procházející skrze logická hradla.

Definice 3. **Tensor**. Tensorový součin $\mathbf{A} \otimes \mathbf{B}$ dvou Chu space $\boldsymbol{A} = (A, r, X)$ a $\boldsymbol{B} = (B, s, Y)$ je definován jako $(A \times B, t, F)$, kde $F \subset Y^A \times X^B$ je množina všech dvojic (f, g) funkcí $f: A \to Y$, $g: B \to X$, pro které $s\big(b, f(a)\big) = r\big(a, g(b)\big)$ pro všechna $a \in A$ a $b \in B$, a $t: (A \times B) \times F$ je dáno $t\big((a, b), f\big) = s\big(b, f(a)\big)\ \Big(= \mathrm{r}\big(\mathrm{a}, \mathrm{g}(\mathrm{b})\big)\Big)$.

3.1.2 Sčítání

Operace sčítání (definice 4) rovněž převzatá z lineární logiky znamená v procesní algebře nezávislou (vzájemně se neovlivňující) paralelní kompozici neboli souběh procesů.

Definice 4. **Sčítání**. Sčítání $\mathbf{A} \oplus \mathbf{B}$ je definováno jako $(A + B, t, X \times Y)$, kde $A + B$ je disjunktní sjednocení množin A a B zatímco $t\big(a, (x, y)\big) = r(a, x)$ a $t\big(b, (x, y)\big) = s(b, y)$.

3.1.3 Zřetězení

Operace zřetězení (definice 5) je definovaná v procesní algebře v (8). Tato operace předpokládá, že abeceda Σ je částečně uspořádaná. Například $0 \leq 1$ pro dvoustavové aktivity nebo $0 \leq 1 \leq 2$ pro třístavové aktivity (respektive $nezačala \leq probíhá \leq doběhla$). To zahrnuje běžné částečné bodové uspořádání Σ^A: dané $f, g: A \to \Sigma, f \leq g$ právě, když $f(a) \leq g(a)$ pro všechna $a \in A$. Záměrem tohoto uspořádání je zavést pojem času, který předepisuje, že v případě stavu $s_0 \in \Sigma$ může přejít do stavu s_1 pouze tehdy, pokud $s_0 \leq s_1$.

Definice 5. **Zřetězení**. Zřetězení dvou Chu space $\mathbf{A}; \mathbf{B}$ je kvocientem $\mathbf{A} \oplus \mathbf{B}$ získaným ponecháním pouze takových stavů (x, y) z $\mathbf{A} \oplus \mathbf{B}$, že buď x je maximální (konečný) stav **A**, nebo y je minimální (počáteční) stav **B** na základě odvozeného částečného bodového uspořádání.

Základní myšlenkou je, že **B** nesmí opustit kterýkoli svůj počáteční stav, dokud **A** nedosáhne některého ze svých konečných stavů.

3.1.4 Výběr

Operace výběr (definice 6) předpokládá, že abeceda Σ obsahuje disjunktivní prvek 0 odpovídající skutečnosti, aktivita ještě nezačala.

Definice 6. **Výběr**. Pokud existuje rozlišovací prvek $0 \in \Sigma$ odpovídající neprovedené události, pak výběr $\mathbf{A} \sqcup \mathbf{B}$ je definován jako $(A + B, t, X + Y)$, kde $t(a, x) = r(a, x)$, $t(b, y) = s(b, y)$ a $t(a, y) = r(b, x) = 0$.

Jinými slovy jde o to, že pokud při operaci výběr **A** ⊔ **B** nastane některý ze stavů **A**, nemůže již být dosažen žádný ze stavů **B** a naopak.

3.2 Vícedimenzionální automat jako Chu space

Následující popis čtyř vybraných operací procesní algebry vícedimenzionálních automatů reprezentovaných jako Chu space je citován z (1) a je také vysvětlován v (10). Základním předpokladem, aby mohly být vícedimenzionální automaty reprezentovány pomocí Chu space, je použití abecedy $\{0,1,2\}$.

Jakýkoliv vícedimenzionální automat s n aktivitami je vlastně částí *n*-dimenzionální krychle. Jelikož víme, že *n*-dimenzionální krychle se skládá z 3^n částí (vrcholů, hran, ploch, a dalších *n*-dimenzionálních objektů), pak všechny tyto části mohou být popsány *n*-ticí znaků z abecedy $\{0,1,2\}$. Tyto *n*-tice pak tvoří sloupce Chu space a odpovídají stavům vícedimenzionálního automatu čili stavům popisovaného procesu. Každý prvek *n*-tice přísluší jedné aktivitě, kde jednotlivé znaky abecedy znamenají:

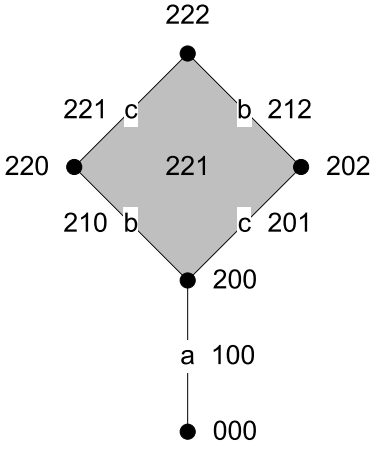
0 – aktivita ještě nebyla provedena,

1 – aktivita se právě provádí,

2 – aktivita byla dokončena.

Příklad reprezentace vícedimenzionálního automatu pomocí Chu space je na obr. 3.1.

Díky tomuto popisu lze v každém stavu jednoznačně říci, které aktivity už byly provedeny, které se právě provádějí, nebo které ještě mohou být provedeny, a samozřejmě můžeme lehce určit část *n*-dimenzionální krychle, která tento stav reprezentuje.



		Stavy									
Aktivity	**a**	0	1	2	2	2	2	2	2	2	2
	b	0	0	0	1	2	0	0	1	2	1	2
	c	0	0	0	0	0	1	2	1	1	2	2

Vícedimenzionální automat **Chu space**

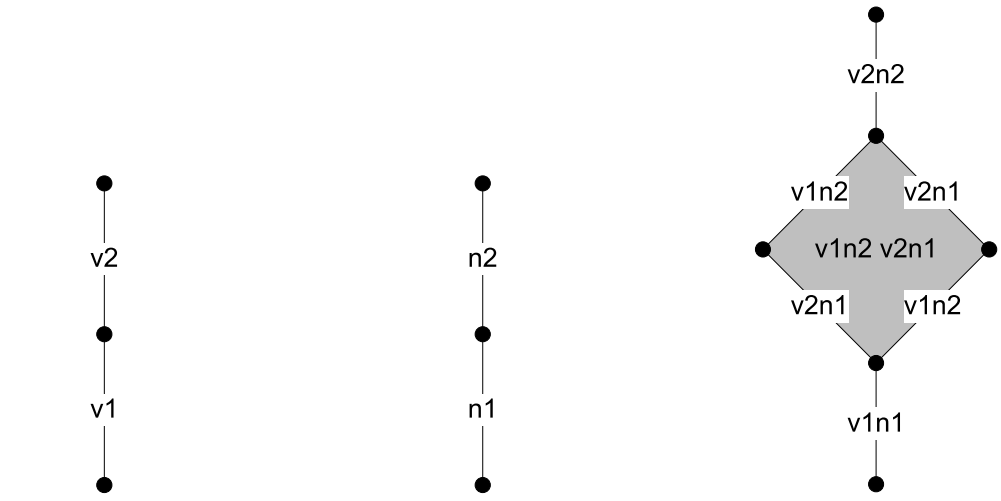
Obr. 3.1: Vícedimenzionální automat reprezentovaný pomocí Chu space

Díky Chu space, které mohou reprezentovat vícedimenzionální automat, získáme operace definované v teorii Chu space. Tyto operace pak reprezentují kompozice jednotlivých procesů. Díky obecnosti Chu space, se dají některé základní operace s Chu space beze změny převést a použít jako operace procesní algebry vícedimenzionálních automatů, jiné zase potřebují jen drobné upřesnění pro použití nad abecedou $\{0,1,2\}$.

Tensorový součin ($\otimes$) z definice 3 může být převzat beze změny. Tensorový součin (orthocurrence), jejíž český překlad by mohl znít „řádný výskyt“, není běžně používanou operací pro kompozici procesů. Význam této operace v procesní algebře není ještě úplně zřejmý, ale V. R. Pratt v (9) a (10) nabízí význam ve smyslu „proplout skrz“. Mějme například dva vlaky jedoucí za sebou po stejné koleji (nemohou se tedy předjet) a dvě nádraží na této trati (obr. 3.2). Pokud vyjádříme tyto dva vlaky za sebou procesem se dvěma sekvenčními aktivitami a dvě za sebou ležící nádraží obdobným procesem se dvěma sekvenčními aktivitami, pak tensorový součin těchto dvou procesů, neobsahuje původní aktivity, ale jejich kombinace, tedy pro příklad s vlaky bude tensorový součin obsahovat aktivity odpovídající výskytům vlaků na jednotlivých nádražích:

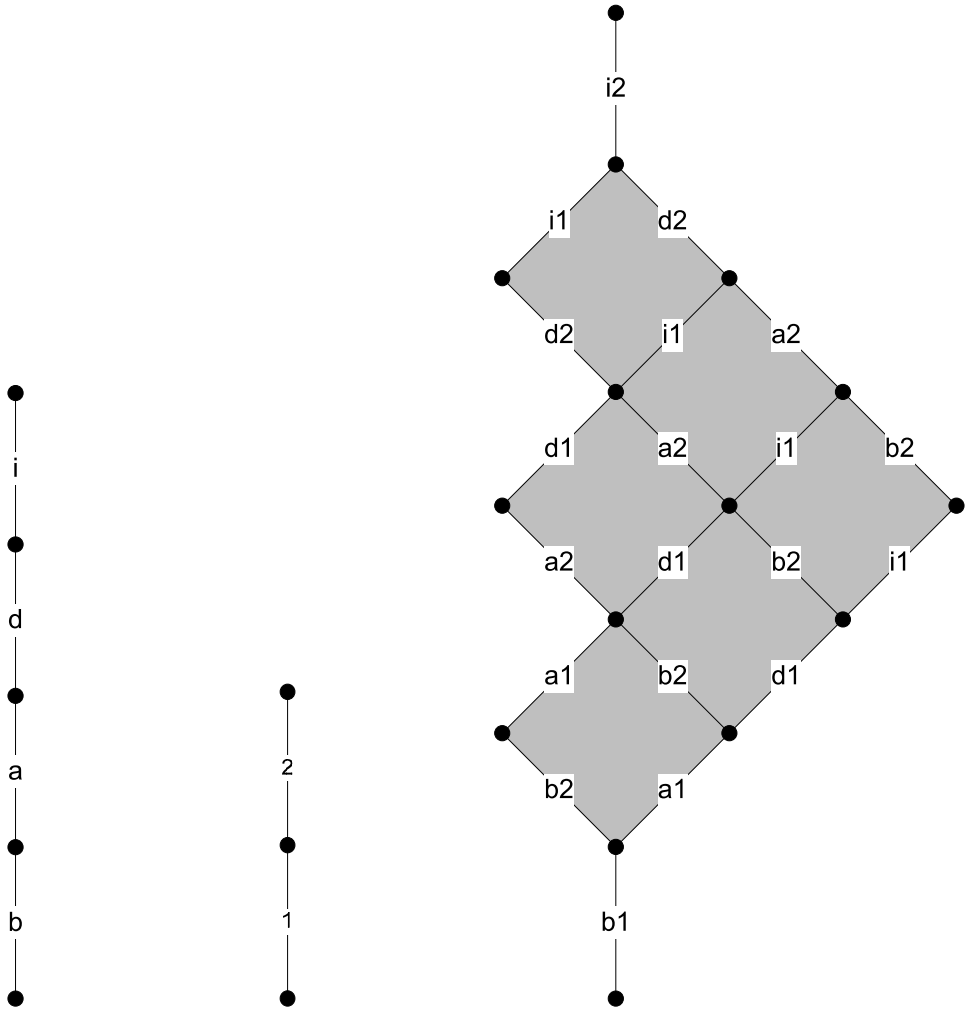
- vlak1 je na nádraží 1,
- vlak1 je na nádraží 2,
- vlak2 je na nádraží 1,
- vlak2 je na nádraží 2.

To lze také chápat jako fakt, že jeden z automatů udává popis procesu (definici procesu), a druhý z nich pak představuje instance těchto procesů, které budou prováděny. V případě sekvenčního uspořádání prováděných procesů, se jejich aktivity nesmějí navzájem „předběhnout“.



Obr. 3.2: Ukázka tensorového součinu pro příklad s vlaky

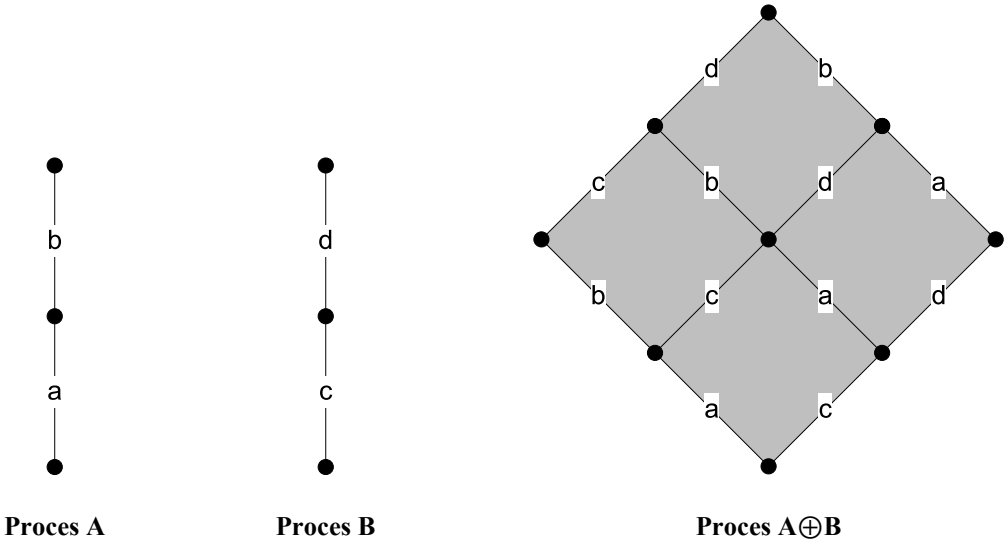
Dalším příkladem takovéto kombinace procesů a jeho instancí může být softwarový proces, který je prováděn v několika iteracích. Softwarový proces se dělí do několika po sobě následujících fází (pro zjednodušení – modelování podnikových procesů, analýza, návrh a implementace). Tyto fáze se provádějí v několika iteracích, jež se navzájem nemohou předbíhat, a také posloupnost softwarového procesu musí být zachována. Tensorový součin pak odpovídá tomu, jak by měly být zpracovávány jednotlivé iterace fází softwarového procesu (obr. 3.3).



Softwarový proces (A) – BPM (b), analýza (a), návrh (d), implementace (i) **Iterace (B)** **Vývoj softwarového produktu ve dvou iteracích (A⊗B)**

Obr. 3.3: Vývoj softwarového produktu ve dvou iteracích jako výsledek tensorového součinu

Operace sčítání (⊕) z definice 4 může být také převzata beze změny a reprezentuje souběh. Příklad paralelní kompozice dvou procesů je na obr. 3.4.



Obr. 3.4: Příklad paralelní kompozice procesů A a B pomocí operace sčítání

Operace zřetězení (;) z definice 5 a výběr (⊔) z definice 6 nelze převzít bez upřesnění. Je totiž třeba definovat potřebné vlastnosti abecedy $\{0,1,2\}$ použité pro vícedimenzionální automaty.

Pro operaci zřetězení je třeba definovat částečné uspořádání nad množinou $\{0,1,2\}$ jako $0 \leq 1 \leq 2$. Pak může být použita operace zřetězení z definice 5. Ta ale pro správnou funkčnost vyžaduje, aby každý koncový stav automatu byl zároveň stavem maximálním, což nemusí být pro podnikové procesy vždy pravda (obr. 3.5).

Existuje ale jiná definice zřetězení, uvedená v definici 7 citované z (10), která nevychází z maximálních a minimálních stavů automatu. Ta vyžaduje množiny počátečních a koncových stavů automatu. Toto pojetí mnohem lépe reflektuje potřeby procesního modelování. Množina koncových stavů je součástí definice vícedimenzionálních automatů stejně jako počáteční stav. Ten může být použit jako jednoprvková množina počátečních stavů. Není tedy třeba dalších úprav, neboť není zvykem při modelování podnikových procesů využívat procesy s více počátečními stavy. V implementaci aplikace ovšem použijeme operaci zřetězení z definice 5, neboť regulární výraz definující kompozici procesu nebude zadávání množiny koncových stavů podporovat.

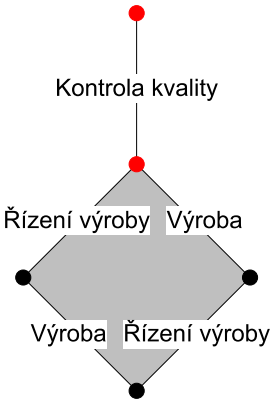
Definice 7. **Zřetězení**. Pro $\mathbf{A} = (A, r, X)$, $\mathbf{B} = (B, s, Y)$, podmnožinu $I_{\mathbf{B}} \subset Y$ počátečních stavů $\mathbf{B}$ a podmnožinu $F_{\mathbf{A}} \subset X$ koncových stavů $\mathbf{A}$ je zřetězení $\mathbf{A};\mathbf{B}$ definováno jako $\mathbf{A};\mathbf{B} =$

$(A + B, t, Z)$, kde $Z \subseteq X \times Y$ je složeno z takových stavů (x, y) pro něž, buď $x \in F_{\mathbf{A}}$ nebo $y \in I_{\mathbf{B}}$ a

$t\big(a, (x, y)\big) = r(a, x)$ pro $(x, y) \in Z$,

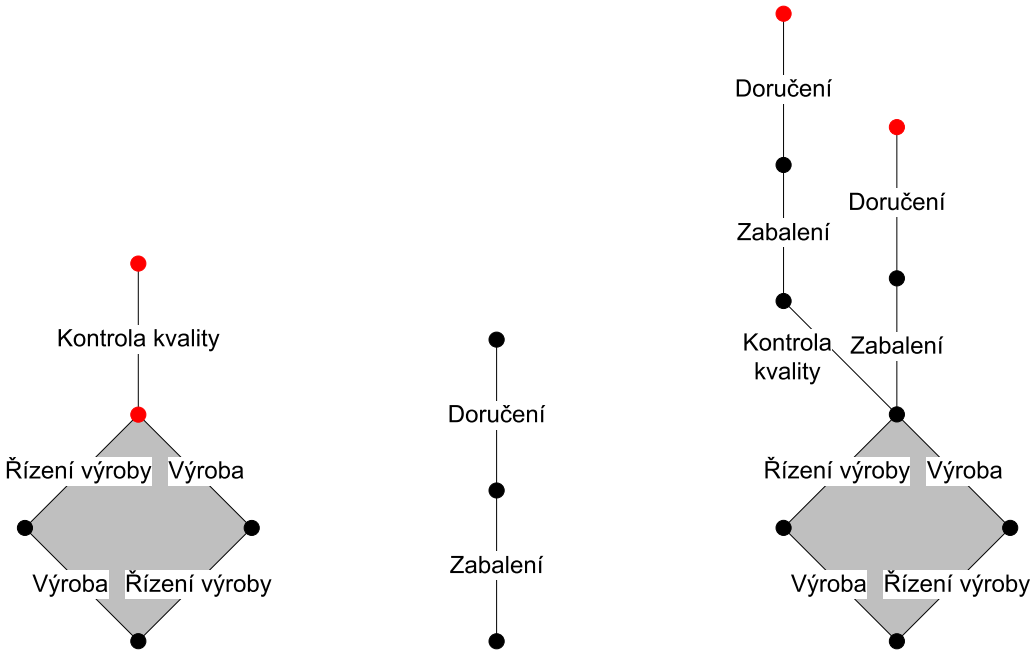
$t\big(b, (x, y)\big) = s(b, y)$ pro $(x, y) \in Z$,

$\mathrm{I}_{\mathbf{A};\mathbf{B}} = \mathrm{Z} \cap (\mathrm{I}_{\mathbf{A}} \times \mathrm{Y})$ a $\mathrm{F}_{\mathbf{A};\mathbf{B}} = \mathrm{Z} \cap (\mathrm{X} \times \mathrm{F}_{\mathbf{B}})$.



Obr. 3.5: Proces se dvěma koncovými stavy (červená kolečka), kde jeden z nich není maximálním stavem automatu

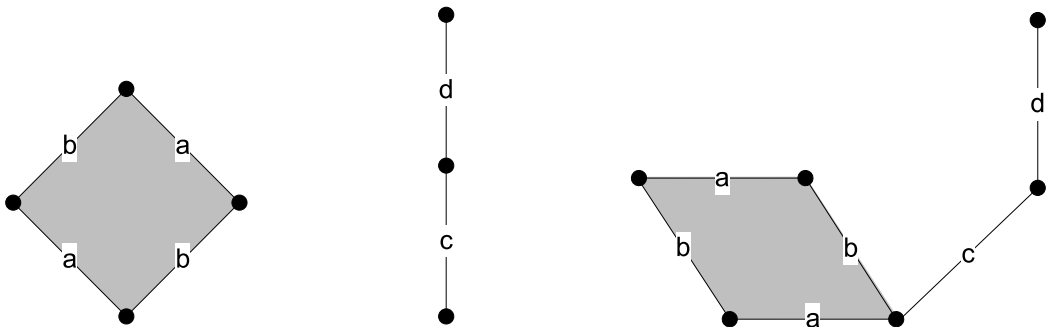
Příklad zřetězení dvou procesů, kde první z nich obsahuje koncový stav, který není maximálním stavem vícedimenzionálního automatu, je na obr. 3.6. Jedná se o proces, kde nejprve souběžně probíhá výroba produktu (Výroba) a řízení výroby (Řízení výroby). Poté následuje kontrola kvality (Kontrola kvality), která ovšem probíhá pouze namátkově v některých případech. Druhým procesem je pouze sekvenční kombinace zabalení produktu k odeslání (Zabalení) a samotné doručení produktu zákazníkovi (Doručení).



Proces A – Výroba produktu **Proces B – Doručení produktu zákazníkovi** **Proces A;B**

Obr. 3.6: Příklad zřetězení dvou procesů A a B

Definice operace výběr vyžaduje speciální prvek abecedy, který oddělí jednotlivé větve procesu a vyznačuje, že aktivita ještě neproběhla. Takovýmto prvkem je v abecedě {0,1,2} prvek 0, který splňuje požadavky kladené v definici 6. Příklad použití této operace je na obr. 3.7.



Proces A **Proces B** **Proces A⊔B**

Obr. 3.7: Příklad kompozice procesů A a B pomocí operace výběr

4 Specifikace požadavků

Jednou z důležitých činností při zadání projektu je specifikování požadavků na samotnou aplikaci (systém). Nedílnou součástí specifikace je vyjmenování hlavní a vedlejší funkcionality aplikace, vytyčení nefunkčních požadavků spjatých s nasazením systému do produkce, specifikace předpokladů, závislostí a omezení kladených na výsledný produkt.

4.1 Funkční požadavky

Ze zadání vzešly základní požadavky na funkcionalitu aplikace:

- vytvoření projektu spravujícího podnikové procesy,
- vytváření podnikových procesů v projektu,
- grafické zobrazení vytvořeného podnikového procesu,
- editování podnikového procesu,
    - editování výrazu definující proces,
    - editování grafického zobrazení procesu,
- smazání podnikového procesu,
- uložení projektu,
- otevření (načtení) projektu,
- uložení grafického zobrazení procesu jako obrázek,
- zobrazení Chu space modelu procesu,
- další pomocné grafické a editační funkce,
    - zobrazení miniatury grafického zobrazení procesu,
    - funkce „zoom“,
    - funkcionalita „zpět/znovu“.

4.2 Nefunkční požadavky

Aplikace bude spouštěna na operačním systému Microsoft Windows verze XP a vyšší.

4.3 Předpoklady a závislosti

Předpokládá se, že aplikace bude co nejméně chybová a bez kritických defektů zapříčiňujících její nefunkčnost. Z pohledu závislostí aplikace nevyužívá připojení do sítě internet, ani některého z databázových systémů.

4.4 Omezení

Omezení aplikace může nastat z hlediska výkonu při grafických operacích. Anebo při náročnějších výpočtech kompozice procesů, zejména při výpočtu modelu pro souběh většího množství aktivit či procesů.

5 Analýza řešení

Z požadavků kladených na funkcionalitu aplikace je zřejmé, že bude poskytovat přehledné grafické uživatelské prostředí, podporovat mnohé grafické operace a bude schopna práci v ní vytvořenou ukládat a znovu načítat. Toto vede k volbě platformy pro implementaci, která bude popsána v kapitole 6. Pro digramy popisující chování nebo statickou strukturu aplikace byl použit jazyk UML.

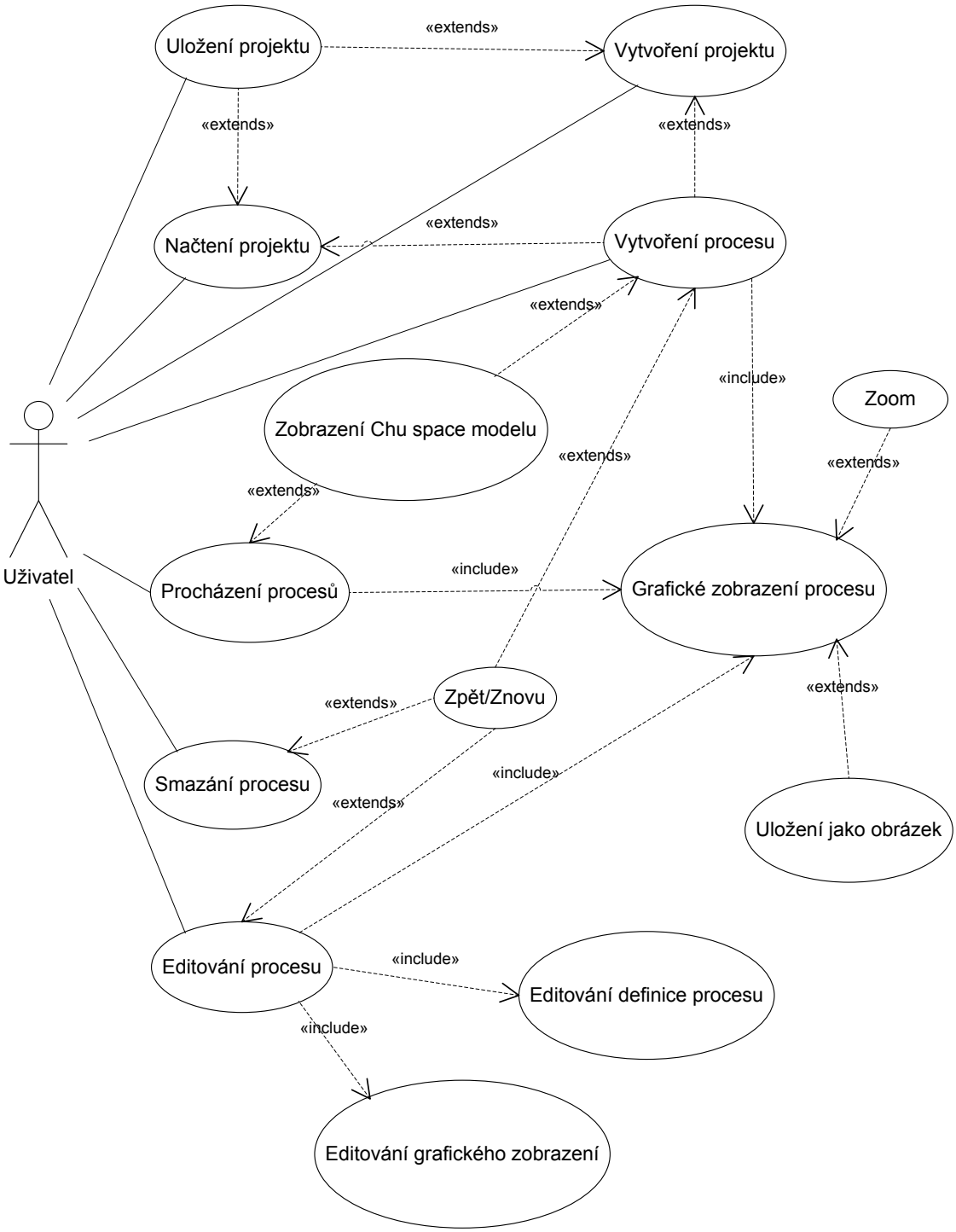
5.1 Diagram případů užití

Součástí analýzy je diagram případů užití zachycující používání aplikace z pohledu uživatelů na obr. 5.1. V aplikaci figuruje jeden aktér (role uživatele), takže každý uživatel má přístup ke všem funkcím systému tomuto aktérovi dostupným.

5.1.1 Popis objektů

Uživatel – reprezentuje aktéra, který aplikaci BPD používá.

Vytvoření projektu – vytvoří projekt aplikace pro správu podnikových procesů.
Uložení projektu – uloží rozpracovaný projekt aplikace.
Načtení projektu – načte projekt aplikace ze souboru na disku.
Vytvoření procesu – vytvoří podnikový proces v příslušném projektu aplikace.
Procházení procesů – umožňuje prohlížet podnikové procesy vytvořené v daném projektu aplikace.
Grafické zobrazení procesu – zobrazí grafickou reprezentaci podnikového procesu.
Zoom – umožňuje zvětšování a zmenšování grafického zobrazení podnikového procesu.
Uložení jako obrázek – uloží grafické zobrazení podnikového procesu jako soubor obrázku.
Zobrazení Chu space modelu – zobrazí matici Chu space podnikového procesu.
Smazání procesu – odstraní podnikový proces v příslušném projektu aplikace.
Editování procesu – provede změny v podnikovém procesu.
Editování definice procesu – edituje jméno nebo regulární výraz definující podnikový proces.
Editování grafického zobrazení – edituje grafickou reprezentaci podnikového procesu.
Zpět/Znovu – provede příkaz „zpět“ nebo „znovu“ na editační úkon, který je právě v pořadí pro daný příkaz.



Obr. 5.1: Diagram případů užití

5.2 Třídní diagram

Dalším diagramem je třídní diagram zachycující datový model pro ukládání, načítání a práci s projektem aplikace na obr. 5.2. Základem datového modelu je objekt sdružující podnikové procesy v konkrétním projektu. Z pohledu modelu vícedimenzionálního automatu jako Chu space a jeho grafické reprezentace by se dal datový model rozdělit na dvě části. První část udržuje procesy v projektu a aktivity a stavy procesu. Druhá část uchovává model grafické reprezentace procesu. Zde jsou zaznamenány grafické objekty pro vizualizaci procesu.

5.2.1 Popis objektů

BpdProject – Kořenový objekt datového modelu. Uchovává seznam podnikových procesů v projektu.

Hda – Reprezentuje podnikový proces. Uchovává seznam aktivit, stavů a grafických prvků *0*-dimenzionálních částí vícedimenzionálního automatu.

Atributy
name – jméno procesu
regExp – regulární výraz definující proces
error – příznak signalizující zda atributy *name* a *regExp* prošly validací
step – základní hodnota vzdálenosti mezi grafickými objekty reprezentující *0*-dimenzionální části vícedimenzionálního automatu použitá pro automatické generování automatu

Activity – Reprezentuje aktivitu podnikového procesu.

Atributy
name – jméno aktivity

State – Reprezentuje stav podnikového procesu jako stav vícedimenzionálního automatu.

Atributy
activityStates – pole hodnot z abecedy $\{0, 1, 2\}$ jednoznačně identifikující stav vícedimenzionálního automatu

Operace
isNode() – zjistí, zda daný stav reprezentuje *0*-dimenzionální část automatu
getDimension() – zjistí číslo dimenze části automatu, kterou daný stav reprezentuje
getActivityStateNumber(ActivityState) – vrátí počet neběžících, běžících nebo dokončených aktivit dle parametru

HdaState – Objekt grafické reprezentace stavu vícedimenzionálního automatu. Slouží k zobrazení objektu *State*, na který má referenci.

Atributy
id – jednoznačný identifikátor grafického prvku
layout – velikost a poloha grafického prvku
name – jméno grafického prvku

Node – *0*-dimenzionální prvek grafické reprezentace vícedimenzionálního automatu.

Operace
addConnection(Edge) – spojí daný *0*-dimenzionální prvek s *1*-dimenzionálním
removeConnection(Edge) – odstraní spojení s *1*-dimenzionálním prvkem

Edge – *1*-dimenzionální prvek grafické reprezentace vícedimenzionálního automatu.

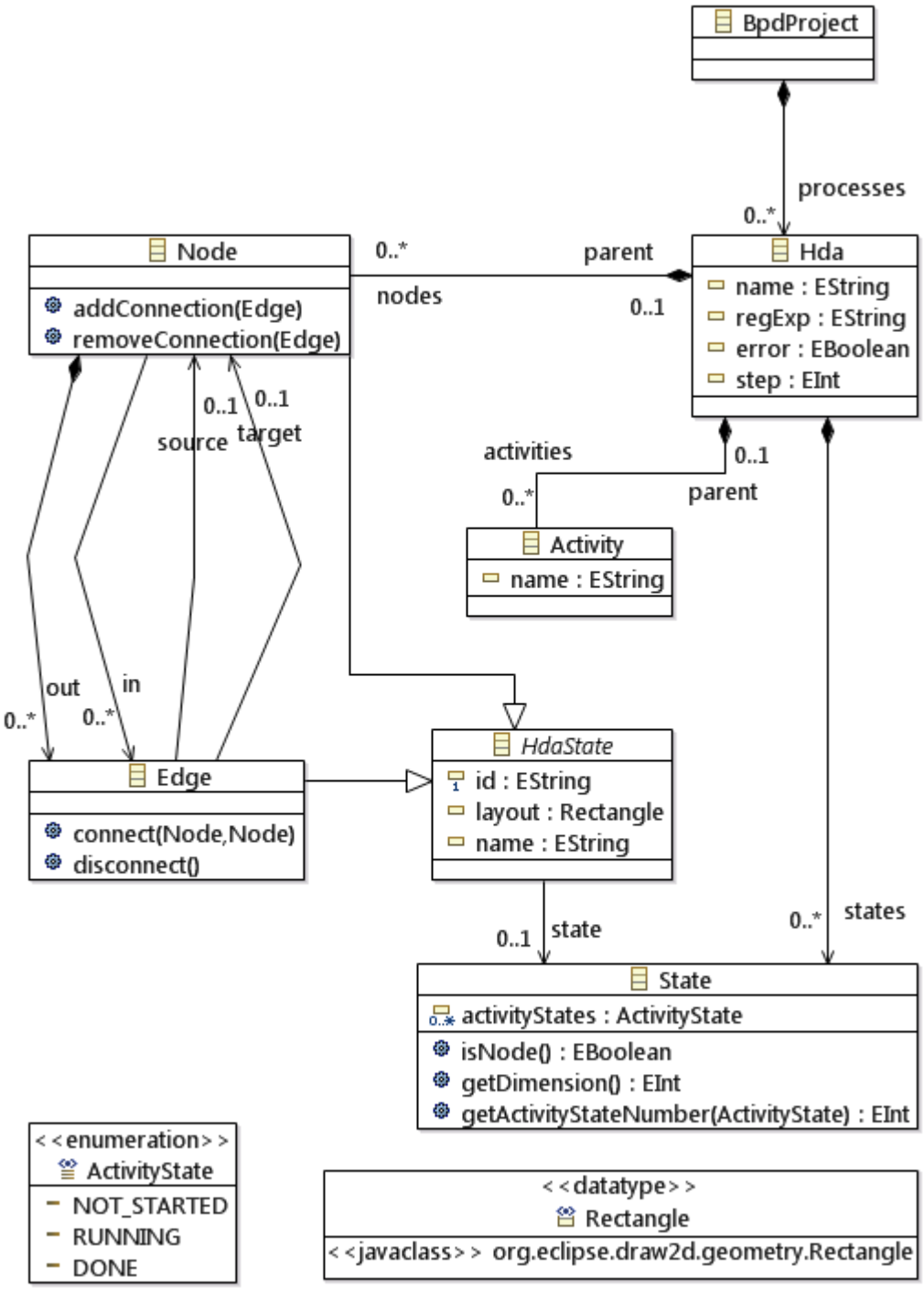
Operace
connect(Node, Node) – propojí dva *0*-dimenzionální prvky
disconnect() – odstraní propojení dvou *0*-dimenzionálních prvků

Rectangle – Definice objektu nesoucí velikost a polohu grafických prvků.

ActivityState – Výčet definující hodnoty abecedy $\{0, 1, 2\}$ pro vícedimenzionální automat reprezentovaný jako Chu space. 0 - *NOT_STARTED*, 1 - *RUNNING*, 2 - *DONE*.

5.3 Název aplikace

Jako název aplikace byl zvolen „Business Process Designer“ (BPD). První dvě slova názvu vznikla přeložením slovního spojení „podnikový proces“ do anglického jazyka. Poslední slovo znamenající „návrhář, konstruktér“, vzešlé z anglického „design“, dodává aplikaci významu jako nástroji pro konstrukci podnikových procesů.



Obr. 5.2: Třídní diagram datového modelu projektu aplikace

6 Použitá technologie

Klíčovým rozhodnutím před samotnou implementací je volba platformy, na které se bude aplikace vyvíjet. Je třeba si uvědomit všechny zásadní požadavky, jež bude muset systém řešit. Zjistit, zda neexistuje platforma nebo aplikační rámec, který již přináší podporu těchto řešení. A jestli tato podpora nepřináší nějaké omezení, kterým se chceme raději vyhnout.

V případě aplikace BPD (Business Process Designer) jsou klíčovými požadavky pro volbu vývojové platformy následující:

- obsluha grafických uživatelských událostí,
- správa $\langle 0; n \rangle$ načtených souborů projektů aplikace,
- podpora operací „zpět“ a „znovu“,
- ukládání a načítání datových modelů do souboru,
- syntaktická analýza regulárních výrazů podnikových procesů,
- sestavení jako samostatné aplikace.

Pro vývoj aplikací jsou v dnešní době nejrozšířenější platformy Java vyvinutá firmou Sun Microsystems a .NET od společnosti Microsoft. Každá z nich poskytuje širokou škálu grafických uživatelských prvků (jak od samotného tvůrce platformy, tak i dalších firem), práci s daty včetně datového formátu XML atd. Pro platformu Java se však nabízí využití tvorby samostatných aplikací na platformě Eclipse, která je na Javě založená. Eclipse totiž nabízí všechny výhody Javy a navíc poskytuje další funkcionalitu a návrhové vzory znovu využitelné při vývoji. Mezi takové rozšíření například patří tyto:

- řešení rozvržení uživatelského rozhraní aplikace pomocí tzv. pohledů „view“ a „editorů“,
- silná podpora podprocesů běžících na popředí i na pozadí,
- nativní grafické komponenty nezávislé na operačním systému,
- aplikační rámec pro tvorbu strukturovaných datových modelů,
- aplikační rámec pro tvorbu grafických editorů z existujících aplikačních modelů,
- vlastní systém nápovědy,
- podpora sestavení výsledné aplikace.

Platforma Eclipse tedy splňuje veškeré hlavní požadavky aplikace BPD. Výhodou je i to, že poskytuje své vývojové prostředí Eclipse IDE, které je v této době k dispozici ve verzi 3.5.2 s pracovním názvem Galileo a ve vývoji je verze 3.6.0 nesoucí pracovní název Helios.

Poslední nevyřešenou otázkou zůstává syntaktická analýza regulárních výrazů podnikových procesů. Ty budou mít v podstatě formát matematického výrazu, a tudíž lze využít teorii gramatik. Pro programovací jazyk Java existuje velké množství generátorů syntaktických analyzátorů založených na gramatikách jako například ANTLR, CUP, JavaCC, Grammatica atd. Z těchto nástrojů byl zvolen právě první jmenovaný ANTLR zdokumentovaný v (11) a to pro svou přehlednost návrhu gramatik a možnost vkládat vlastní zdrojové kódy do definice

gramatiky. ANTLR také poskytuje nástroj ANTLRWorks nebo rozšíření Eclipse IDE v podobě modulu pro tvorbu gramatik.

6.1 Eclipse

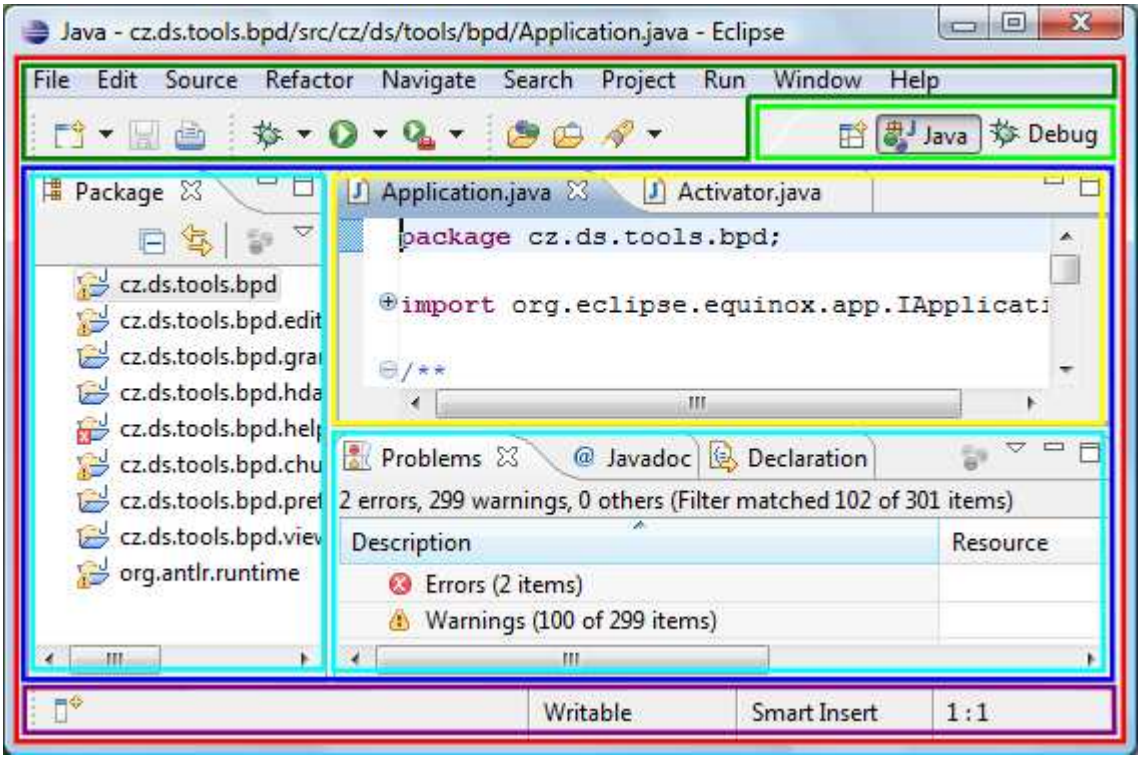
Eclipse je skupina lidí, která se zabývá vývojem nástrojů založených na platformě Java s otevřeným zdrojovým kódem. Nejznámějším výstupem této skupiny je integrované vývojové prostředí Eclipse IDE (Integrated Development Environment). V dnešní době již vývojové prostředí Eclipse poskytuje podporu nejenom pro Javu, ale i další programovací jazyky a jejich nástroje jako C/C++, Python, PHP, webové technologie, manipulace s daty a reportování. Jednotlivé součásti Eclipse lze kombinovat a integrovat dle potřeby. Obecnou platformou pro běh aplikací, na které je postaven i Eclipse IDE je Eclipse RCP (Rich Client Platform).

6.2 Eclipse RCP

Eclipse RCP se začala vyvíjet od verze Eclipse 2.1. Od verze Eclipse 3.0 je plnohodnotnou součástí Eclipse platformy. Eclipse RCP je založena na vlastní implementaci technologie OSGi (Open Services Gateway initiative) nazývanou Equinox. OSGi je dynamický modulární systém pro Javu. Umožňuje instalaci, update a odebírání modulů nazývaných „bundle“ za běhu, definuje životní cyklus modulů a nabízí infrastrukturu pro spolupráci modulů prostřednictvím služeb. Důležitou vlastností je také jednoduché verzování modulů. Ve svém základu obsahuje Eclipse RCP moduly pro uživatelské rozhraní, běhové prostředí a OSGi. Výsledná aplikace je poté sestavena pouze z potřebných modulů (použitých modulů a modulů jejich závislostí). Postup, jak vytvářet Eclipse RCP aplikace je popsán v (12). Podrobným popisem procesu, jak vytvářet vlastní moduly a tím rozšiřovat platformu Eclipse nebo aplikace na ní založené, se zabývá (13).

6.3 Pracovní plocha Eclipse

Základní pracovní plocha Eclipse RCP aplikace se nazývá „workbench“. Pracovní plocha obsahuje lištu menu (menu bar), nástrojovou lištu (tool bar), stavový řádek (status bar) a jednu nebo více perspektiv (perspective). Lišta menu a nástrojová lišta obsahuje příkazy a akce, které může uživatel v aplikaci provést. Stavový řádek slouží pro zobrazování stavových informací aplikace. Perspektiva je grafické rozložení aplikace definující, jaké pohledy (views) a nástroje budou uživateli zobrazeny a ve které části pracovní plochy budou umístěny. Další součástí perspektivy může být oblast editorů, kde se zobrazují otevřené soubory. Editor je speciální případ pohledu určený k prohlížení a editování dat. Jeden z možných případů pracovní plochy Eclipse je znázorněn na obr. 6.1.



Pracovní plocha
Lišta menu, nástrojová lišta
Přepínače perspektiv
Stavový řádek
Oblast pohledů
Pohledy
Editory

Obr. 6.1: Pracovní plocha Eclipse

6.4 Eclipse Modeling Framework

Eclipse Modeling Framework (EMF) je použit pro datový model aplikace. Jednoduše řečeno jde o modelovací aplikační rámec pro Eclipse popsaný v (14), který umožňuje vývojářům rychle budovat robustní aplikace založené na překvapivě jednoduchých modelech. Cílem takto vytvářených modelů je zaměřit se na koncepci modelu jako takového a nikoli na jeho implementační detaily. Klíčovou koncepcí EMF jsou:

- meta-data modelovaných objektů,
- generování kódu modelu,
- výchozí serializace.

Každá specifikace nově vytvářeného modelu je popsána ve formátu XMI[1]. EMF poskytuje nástroje a běhové prostředí k produkování souboru Java tříd pro model, spolu se souborem tříd adaptérů[2], které umožňují prohlížení modelu a jeho editování založené na příkazech[3].

Jak již bylo nastíněno, EMF umožňuje serializaci jednotlivých objektů modelu do XML formátu a naopak. Soubory projektu aplikace BPD budou taktéž ukládány v dnes velmi oblíbeném formátu XML.

Mezi další schopnosti EMF, které stojí za zmínku a nebudou všechny vyjmenovány kvůli obsáhlosti tohoto aplikačního rámce, patří tyto:

- definování modelů pomocí Java rozhraní a anotací, UML, XML schéma nebo vlastním formátem EMF nazývaným Ecore,
- velké množství anotací zjednodušující výsledný kód,
- mapování modelů na relační databáze,
- mechanizmus porovnávání modelů,
- schopnost dotazování se na modelové objekty a jejich obsah,
- podpora transakcí pro správu zdrojů,
- aplikační rámec pro validaci dat.

6.5 Graphical Editing Framework

Graphical Editing Framework (GEF) umožňuje vývojářům vytvářet vlastní grafické editory na platformě Eclipse z existujícího modelu aplikace. Následující přehled je převzat z (15). GEF nabízí mechanizmus pro rozložení a vykreslování zobrazované grafiky. Vývojáři pak mohou využít velké množství grafických operací, které GEF poskytuje anebo je pro specifické případy rozšířit.

GEF používá návrhový vzor Model-View-Cotroller (MVC) vyobrazený na obr. 6.2. MVC rozděluje aplikaci na tři části:

- Model - uchovává a zpracovává data aplikace, zapouzdřuje stav aplikace, reaguje na dotaz na stav.

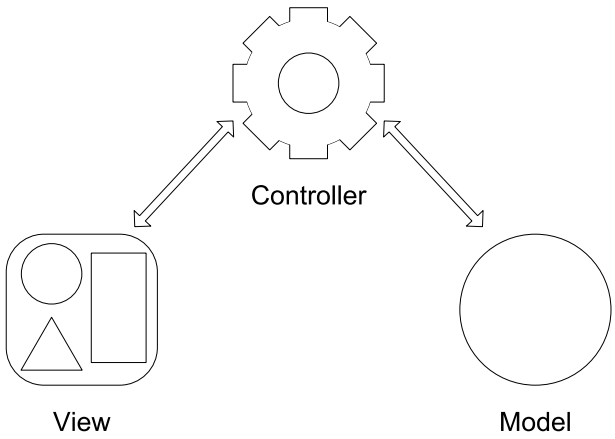
---

[1] Jak se uvádí v (19) XMI je standard pro výměnu informací z meta-dat pomocí jazyka XML. Cílem je rozdělení dat do abstraktního a konkrétního modelu, aby mohlo docházet k výměně informací mezi modely různých nástrojů či aplikací, které XMI využívají. Asi nejčastější použití XMI je jako výměnný formát pro UML modely.

[2] Třída adaptéru – je v programovacím jazyce Java používána jako mezičlánek implementace rozhraní a třídy vytvořené programátorem. Třída adaptéru implementuje všechny metody rozhraní, nejčastěji rozhraní nějakého posluchače. Programátor tedy vytvoří podtřídu třídy adaptéru a přepíše pouze metody, které potřebuje. Zdroj této definice je v (19).

[3] Tzv. „command-based“ editování znamená, že pro jednotlivé editační úkony jsou vytvářeny příkazy, které mohou být provedeny na zásobníku příkazů (z anglického „command stack“). Ze zásobníku příkazů jsou poté volány implementace operací „zpět“ a „znovu“ příslušného příkazu, který je na zásobníku právě v pořadí, podle interakce uživatele.

- View - posílá uživatelské události Controlleru. Žádá Model o aktuální data, získává data a rozhoduje, jak je zobrazit.
- Controller - definuje chování aplikace, spolupracuje s View a konvertuje klientská data tak, aby jim rozuměl Model.



Obr. 6.2: Návrhový vzor Model-View-Controller

Controller je tedy mostem mezi zobrazovací částí a modelem. Každý controller, nebo „EditPart“ jak je označován v GEF, je zodpovědný za mapování modelu na své view a za provádění změn v modelu. EditPart také sleduje model a aktualizuje view tak, aby odráželo změny na modelu. EditPart je objekt, který reaguje na interakce uživatele.

View je reprezentováno zobrazovacím objektem nazývaným „viewer“. GEF poskytuje dva typy: grafický a stromový. V aplikaci BPD bude použit grafický viewer. Jeho podstata je taková, že na plátno vykresluje tvary. Rozhodnutí o tom, který tvar se má vykreslit, zajišťuje opět controller.

6.6 ANTLR

ANother Tool for Language Recognition (ANTLR) je nástroj poskytující aplikační rámec pro konstrukci jazykových rozpoznávačů, překladačů kódu a překladačů gramatik obsahujících příkazy v různých cílových jazycích, jak se uvádí v (16). ANTLR automatizuje konstrukci jazykového rozpoznávače. Z formální gramatiky generuje program, který určuje, zda dané věty vyhovují příslušnému jazyku. Jinými slovy se jedná o program, který píše další programy. Přidáním fragmentů kódu do gramatiky se jazykový rozpoznávač stává překladačem nebo interpretem. ANTLR poskytuje vynikající podporu pro stromové konstrukce, procházení stromem, překlady a poskytuje sofistikované automatické zotavení po chybě a podávání zpráv.

7 Návrh aplikace BPD

V této kapitole je zachycen návrh statické struktury aplikace BDP. Ta bude obsahovat dvě klíčové oblasti. V první z nich bude vytvářeno hlavní okno aplikace, viz obr. 7.1. Druhá se bude starat o zobrazovací část editoru podnikových procesů, správu editačních příkazů a uložení projektu aplikace do souboru viz obr. 7.2. Napojení datového modelu na editor je znázorněno na obr. 7.3.

7.1 Hlavní okno

Hlavní okno aplikace bude vytvářeno ve třídě `cz.ds.tools.bpd.Application` implementující rozhraní `org.eclipse.equinox.app.IApplication`, jejíž metoda `start()` bude volána inicializačním mechanizmem platformy Eclipse. Tato metoda v rámci budování okna uživatelského rozhraní aplikace vytvoří i lištu menu, nástrojovou lištu a pracovní plochu. Metoda `stop()` ve stejné třídě bude volána pro ukončení grafického uživatelského rozhraní při zavírání aplikace.

Třída `cz.ds.tools.bpd.ApplicationWorkbenchAdvisor` rozšiřující abstraktní třídu `org.eclipse.ui.application.WorkbenchAdvisor` a třída `cz.ds.tools.bpd.ApplicationWorkbenchWindowAdvisor` rozšiřující třídu `org.eclipse.ui.application.WorkbenchWindowAdvisor` budou sloužit ke konfiguraci pracovní plochy.

Pro konfiguraci lišty menu a nástrojové lišty bude sloužit třída `cz.ds.tools.bpd.ApplicationActionBarAdvisor`, která rozšiřuje třídu `org.eclipse.ui.application.ActionBarAdvisor`. Zde se také budou vytvářet akce pro menu a rolovací nabídky.

7.2 Editor projektu

Třída `cz.ds.tools.bpd.editor.editors.HdaGraphicalEditor` je základem editoru projektu. O vytváření grafických objektů pro editor se stará tovární třída `cz.ds.tools.bpd.editor.part.HdaEditPartFactory`. Ta podle modelového objektu, který se má zobrazit v editoru, vytvoří příslušný editační objekt. A to `HdaPart` pro základní grafickou vrstvu diagramu, `NodePart` pro počáteční a koncové stavy aktivit a `EdgePart` pro spojnice mezi nimi. Všechny tyto editační objekty jsou ze stejného balíku jako výše zmíněná tovární třída. Přímo pro grafické zobrazení modelového objektu obsahuje každý editační objekt svůj grafický objekt. `HdaPart` je graficky reprezentován třídou `HdaFigure`, `NodePart` třídou `NodeFigure`, obě z balíku `cz.ds.tools.bpd.editor.figure`. `EdgePart` je reprezentován třídou `org.eclipse.draw2d.PolylineConnection`. O změnách na modelu, jsou editační objekty informovány prostřednictvím své nadřazené abstraktní třídy `cz.ds.tools.bpd.editor.part.HdaAbstractEditPart`, která na

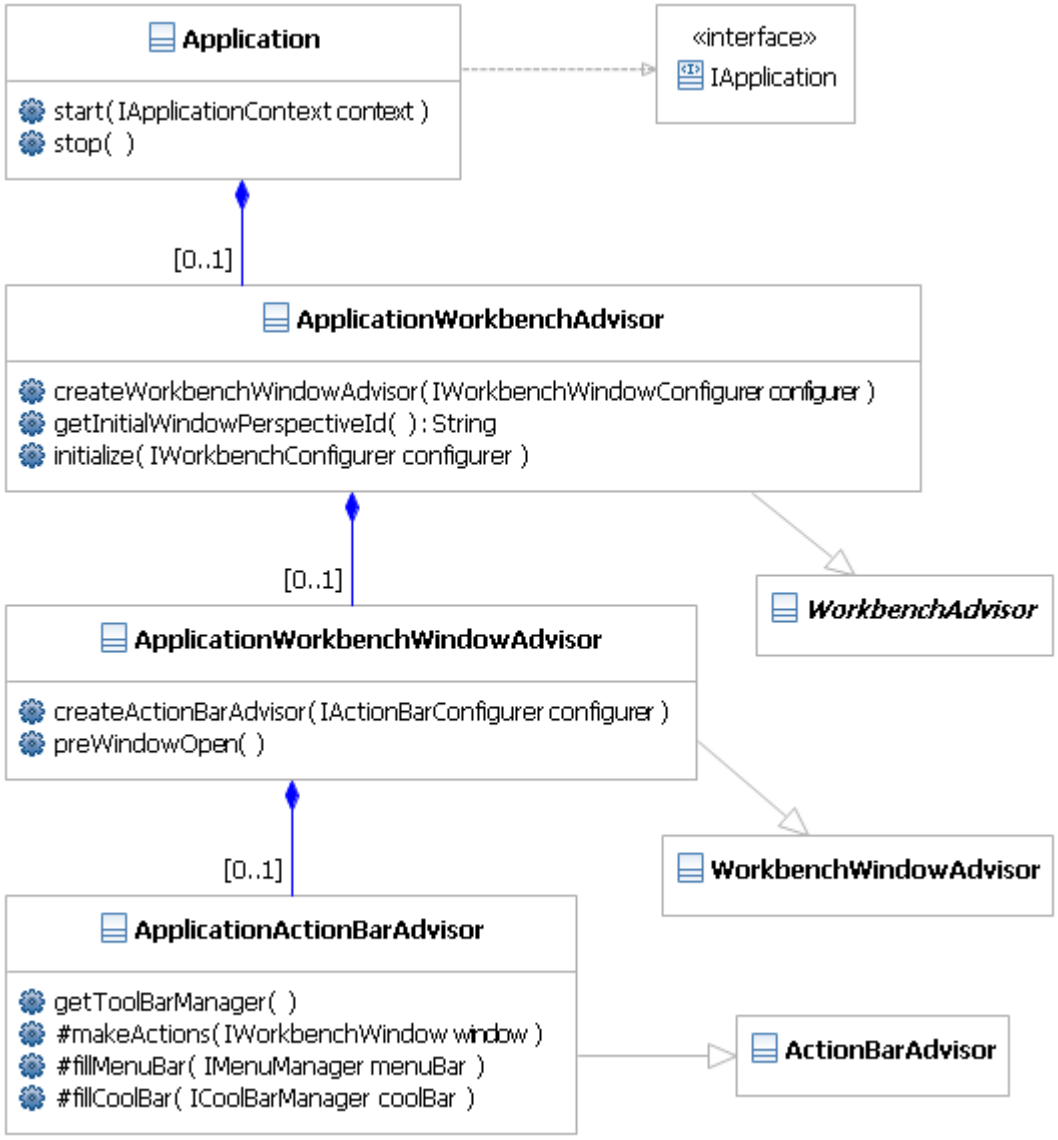
modelu poslouchá. Přidání a odebrání posluchače na model editačního objektu zajišťují metody `activate()` a `deactivate()`, které volá mechanizmus GEF při jejich vytváření a rušení.

Správou editačních příkazů je pověřena editační doména editoru tvořená třídou `org.eclipse.gef.DefaultEditDomain`. Ta obsahuje zásobník příkazů reprezentovaný třídou `org.eclipse.gef.commands.CommandStack`. Na zásobníku příkazů má editor zaregistrovaného posluchače, který musí být objektem implementujícím rozhraní `org.eclipse.gef.commands.CommandStackListener`. Tento posluchač je informován vždy, když dojde ke změně stavu zásobníku příkazů (je proveden některý z editačních příkazů nebo je volána některá z akcí „zpět“ a „znovu“) a podle toho mění stav editoru (zda je v uloženém nebo neuloženém stavu a které akce jsou pro něj aktuálně dostupné).

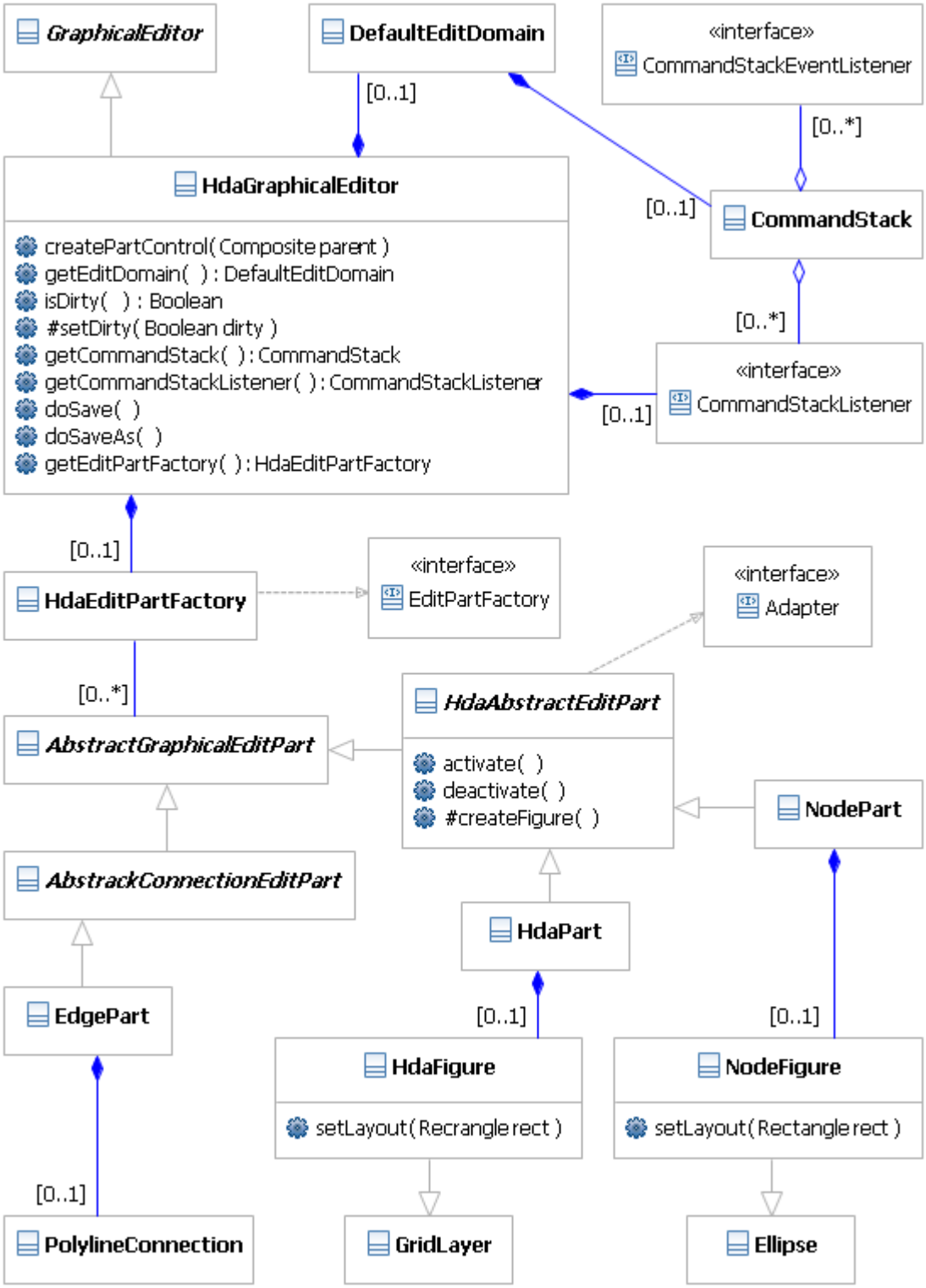
Načtený projekt bude možno uložit pod novým jménem voláním metody `doSaveAs()` ve třídě `cz.ds.tools.bpd.editor.editors.HdaGraphicalEditor` nebo pod stávajícím jménem, pokud již uložen je, voláním metody `doSave()` ve stejné třídě.

7.2.1 Napojení modelu na editor

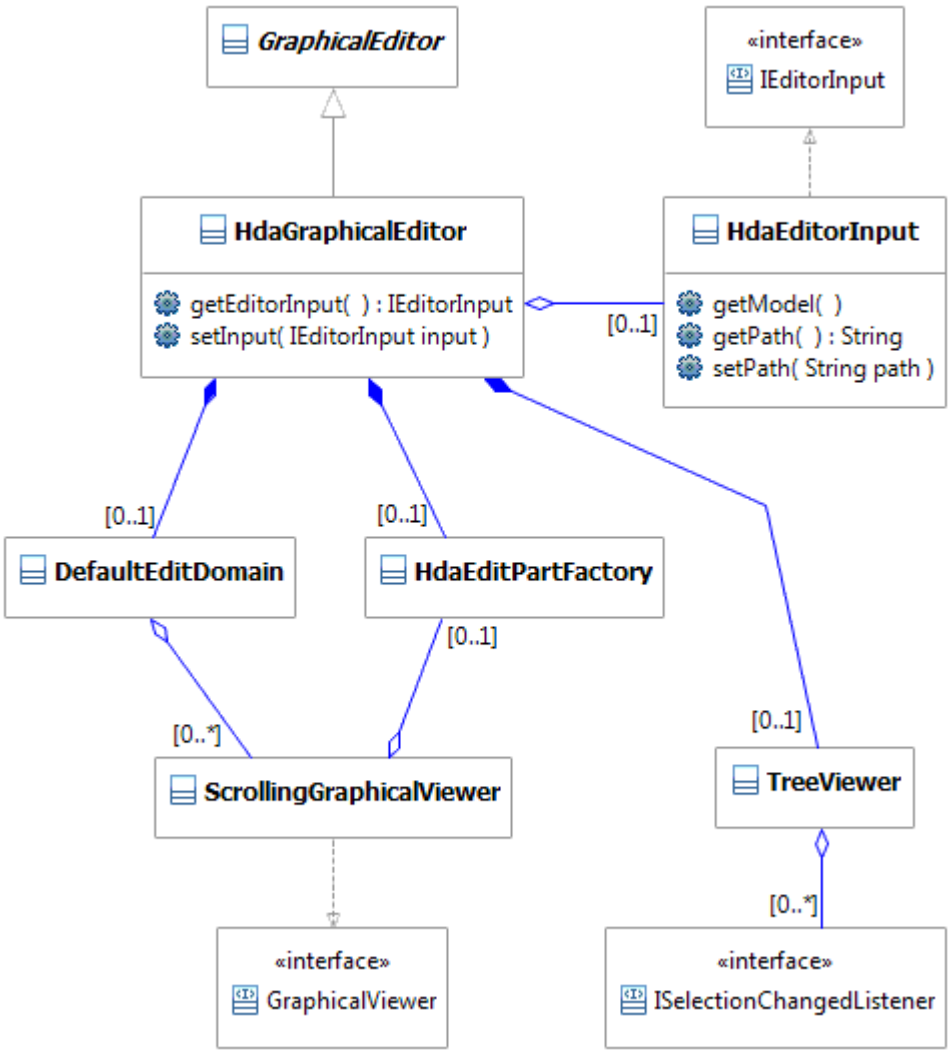
Model je napojen na editor prostřednictvím tzv. vstupu editoru. Vstupem editoru je objekt třídy `cz.ds.tools.bpd.editor.input.HdaEditorInput`, který pro získání modelu obsahuje metodu `getModel()`. Seznam podnikových procesů modelu je poté zobrazen objektem třídy `org.eclipse.jface.viewers.TreeViewer`. V tomto stromovém seznamu vybraný proces je graficky zobrazen objektem třídy `org.eclipse.gef.ui.parts.ScrollingGraphicalViewer`.



Obr. 7.1: Třídní diagram hlavního okna aplikace



Obr. 7.2: Třídní diagram editoru aplikace



Obr. 7.3: Třídní diagram napojení datového modelu na editor

8 Implementace aplikace BPD

Pro vývoj aplikace BPD byla zvolena v té době aktuální verze Eclipse IDE 3.4.2 s pracovním názvem Ganymede. Pro využití modelovacího aplikačního rámce EMF a aplikačního rámce s podporou grafických operací GEF je nutné nainstalovat distribuci nazvanou Eclipse Modeling Tools. Ta obsahuje balíčky výše zmíněných aplikačních rámců i podporu pro vývoj RCP aplikací a mechanismu pro sestavení výsledného programu. Aktuální verze všech distribucí platformy Eclipse je k dispozici ke stažení na (17).

Během vývoje BPD dospěl Eclipse do verze 3.5.2. Přechod na vyšší verzi tohoto prostředí však zpravidla není problém. Eclipse totiž pro správu vyvíjených projektů používá svůj pracovní prostor nazývaný „workspace“. Při změně na vyšší nebo jinou verzi nainstalujeme tu požadovanou, která se napojí na stávající pracovní prostor.

Jak již bylo zmíněno v kapitole 6.2, Eclipse RCP aplikace je složena z modulů označovaných jako „bundle“ a to buď vlastních anebo poskytovaných platformou Eclipse. Tyto moduly mohou být dvou typů: „feature“ a „plugin[4]“. Feature lze chápat jako samostatnou logickou jednotku, která ve své podstatě obsahuje seznam „pluginů“ a jiných feature. Eclipse používá feature jako prostředek pro aktualizace (popřípadě instalování nebo odinstalování) a pro proces sestavování aplikací. S feature můžeme dodávat i programovou licenci. Druhý z modulů, plugin, zapouzdřuje vždy konkrétní funkcionalitu a obsahuje tedy výkonný kód, ikony a jiné zdroje. Pomocí těchto modulů můžeme rozšiřovat stávající funkcionalitu Eclipse IDE a z nich je složena i Eclipse RCP aplikace. Podrobný popis vnitřní struktury modulu je popsán v následující kapitole 8.1.

8.1 Eclipse plugin

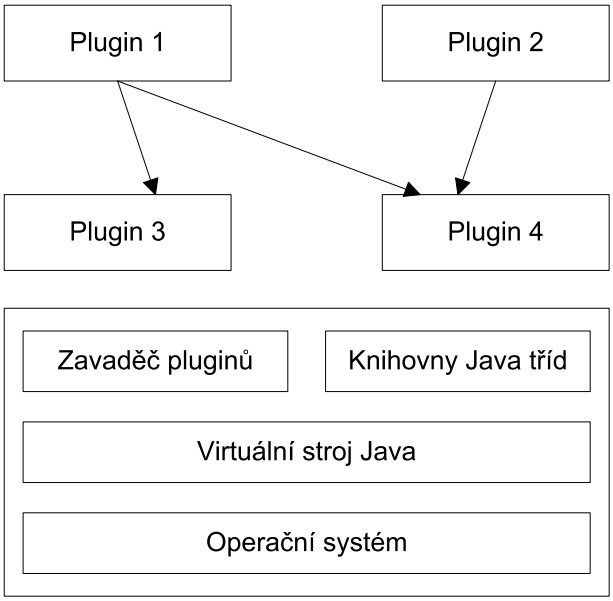
Plugin je výkonný modul Eclipse platformy. Chování každého modulu je v jeho kódu. Jeho závislosti a služby, které poskytuje, jsou deklarovány v souborech `MANIFEST.MF` a `plugin.xml`. Tato struktura usnadňuje postupné načítání kódu modulu ve chvíli, kdy je daný úsek kódu spouštěn, čímž se snižuje doba spouštění a paměťové nároky Eclipse aplikace. Příklad propojení na sobě závislých modulů a jejich zavaděče znázorňuje schéma na obr. 8.1. Je nutné ještě podotknout, že Eclipse nedovoluje cyklické závislosti modulů například dle ukázky na obr. 8.2. Snahou každého vývojáře by mělo být, aby jím vytvořené moduly vždy pokrývaly konkrétní související funkcionalitu. Tím se zamezí problémům týkajících se hierarchizace modulů během vývoje nebo v jejich budoucí integraci. S tím je spojená také konvence dobrého pojmenovávání těchto modulů pro snadnější orientaci mezi nimi.

Zavaděč modulů při startu prozkoumá soubory `MANIFEST.MF` a `plugin.xml` pro každý modul a poté sestaví strukturu z informací, které obsahují. Tato struktura sice zabere část

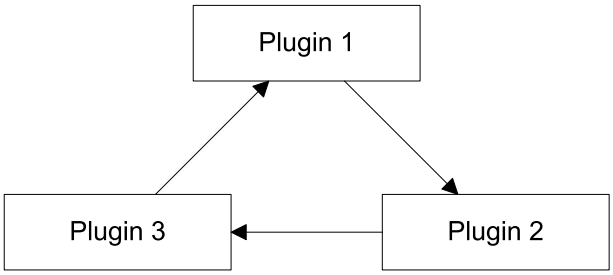
[4] V některých literaturách se můžeme setkat s označením „plug-in“.

paměti, ale to umožňuje zavaděči najít požadovaný modul mnohem rychleji a zabírá výrazně méně paměťového prostoru než načtení všech výkonných kódů ze všech modulů po celou dobu běhu aplikace.

Modul může být distribuován buď v samostatné složce, a to například z důvodu, že obsahuje nějaké dynamicky linkované knihovny. Anebo, jak je tomu většinou, jako komprimovaný JAR soubor.



Obr. 8.1: Struktura závislosti modulů



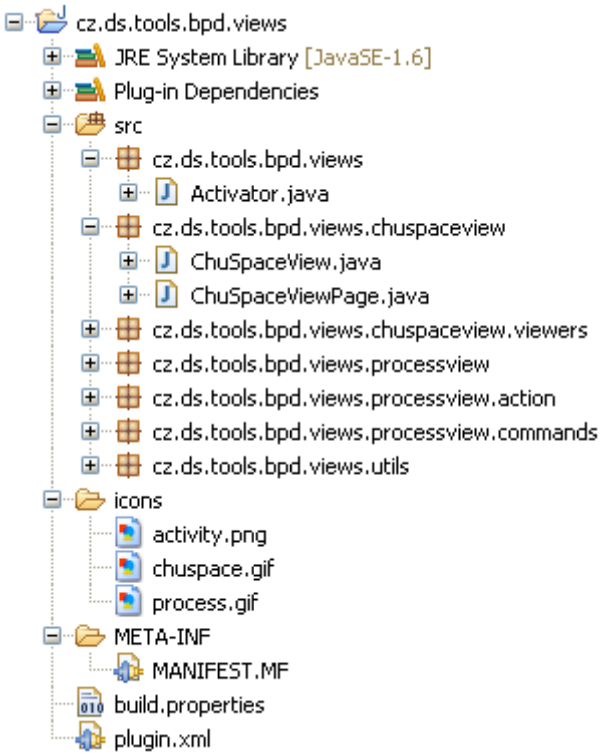
Obr. 8.2: Ukázka cyklické závislosti modulů

8.1.1 Struktura modulu

Základní struktura souborového systému modulu je následující. Konkrétní příklad je uveden na obr. 8.3.

java třídy – jsou umístěny v příslušném java adresáři dle balíčku, ve kterém se nachází.

ikony – soubory obrázků jsou obvykle umístěny v adresáři nazvaném „icons“ nebo „images“. K souborům obrázků a dalším statickým zdrojovým souborům, které jsou dodávány jako součást modulu, lze přistupovat pomocí metod v aktivátoru modulu popsaném v kapitole 8.1.3.
META-INF/MANIFEST.MF – tento soubor popisuje aspekty modulu za běhu programu, jako identifikátor, jméno, verze, jeho závislosti atd.
plugin.xml – soubor ve formátu XML popisující rozšíření o funkcionalitu, kterou navenek vystavuje jiný modul a deklarace funkcionality, kterou vystavuje jiným modulům modul samotný.



Obr. 8.3: Příklad struktury souborového systému modulu

8.1.2 Rozšíření a body rozšíření

Každý modul může deklarovat tzv. „body rozšíření“ (extension points) tak, že ostatní moduly mohou rozšířit funkcionalitu původního modulu. Deklarace bodů rozšíření a definice rozšíření se nachází v souboru `plugin.xml` v kořeni souborového systému modulu. Tento mechanizmus poskytuje oddělující vrstvu takovou, aby původní modul nepotřeboval vědět o existenci modulu, který ho rozšiřuje v okamžiku sestavení původního modulu.

Každý typ bodu rozšíření může k definici rozšíření vyžadovat různé atributy. Mohou to být identifikátory rozšíření, cesty ke zdrojům jako ikonám atd., cesty k třídám implementujícím potřebná rozhraní pro rozšíření a další nastavení.

8.1.3 Aktivátor modulu

Aktivátor je třída modulu, která poskytuje metody pro přístup ke statickým zdrojům v rámci modulu, přístup a inicializaci nastavení modulu a pro informace o stavu. Aktivátor není povinný, ale pokud je uveden v souboru `MANIFEST.MF`, je první třídou, která dostane zprávu hned poté, co byl modul načten. Zároveň je poslední třídou, která je informována o tom, že modul bude odpojen.

8.2 Moduly BPD

Aplikace BPD sestává z několika vlastních modulů a modulů platformy Eclipse, na kterých jsou závislé. Pojmenovávání modulů jsem volil dle doporučení v Eclipse. Jednoznačný identifikátor modulu se skládá z doménových názvů oddělených tečkou. Názvy domén jsou v identifikátoru řazeny od domény nejvyššího řádu zleva, jak je vidět na obr. 8.4.

Obr. 8.4: Ukázka pojmenování modulu

Není předepsáno, kolik řádů by identifikátory měli mít. První dva řády však v naprosté většině mají stejný význam, viz níže. Další řády se mohou lišit v závislosti na autorovi nebo společnosti. Jednotlivé řády domény představují následující oblasti:

1. Rozlišuje, zda se jedná o modul pro komerční využití nebo spadá pod neziskovou organizaci anebo jen určuje zemi, ve které byl vytvořen. V tomto případě je použita zkratka „cz“ označující Českou republiku.
2. Označuje společnost nebo autora, který modul vyvinul. Zkratka „ds“ jsou mé iniciály.
3. Jde o upřesnění podskupiny programů nebo rozšíření programu, do které modul patří. Název „tools“ značí podskupinu programových nástrojů.
4. Zkratka vyvíjené aplikace (popřípadě rozšíření aplikace).
5. Konkrétní funkcionalita, kterou modul pokrývá. Ta může být ještě blíže specifikována v doménách vyšších řádů.

Aplikaci BPD tvoří ve svém základu 9 modulů:

cz.ds.tools.bpd

Hlavní modul aplikace. Obsahuje metody pro spuštění a inicializaci prostředí BPD. Definuje lištu menu, nástrojovou lištu, stavový řádek a výchozí rozložení oken

grafického uživatelského rozhraní. Dále se stará o akce pro vytváření nového nebo otevírání existujícího projektu a o zavření aplikace včetně uložení jejího stavu.

cz.ds.tools.bpd.editor

Zahrnuje kompletní správu nad editorem projektu aplikace. Vytváří editor pro konkrétní projektový soubor a definuje pro něj akci pro vytvoření nového podnikového procesu včetně jejího průvodce, akce pro procházení procesů projektu, odstranění procesu a uložení grafického zobrazení procesu jako obrázek.

cz.ds.tools.bpd.grammar

Obsahuje gramatiku vytvořenou pomocí nástroje ANTLR (kapitola 6.6) pro regulární výraz definující podnikový proces, kódy překladače gramatiky a metody pro vytvoření Chu space modelu z regulárního výrazu procesu na této gramatice. Nedílnou součástí je také gramatika a další validační metody regulárního výrazu procesu zadaného uživatelem.

cz.ds.tools.bpd.hda.model

Definuje model projektu aplikace v aplikačním rámci EMF (kapitola 6.4). Implementuje objekty modelu a tovární metody pro jejich vytváření. Poskytuje metody pro načtení modelu ze souboru a uložení modelu do souboru s využitím výchozí serializace modelových objektů.

cz.ds.tools.bpd.help

Obsahuje uživatelskou dokumentaci a nápovědu aplikace.

cz.ds.tools.bpd.chuspace

Definuje model pro Chu space podnikového procesu včetně základních operací proveditelných nad Chu space. A dále metody pro vytvoření vícedimenzionálního automatu reprezentujícího proces z Chu space.

cz.ds.tools.bpd.preferences

Obsahuje dialogy a mechanizmus pro inicializaci, změnu a uložení předvoleb a nastavení aplikace.

cz.ds.tools.bpd.views

Vytváří pohled (pojem „pohled“ vysvětlen v kapitole 6.3) pro zobrazení a editování definice podnikového procesu a poskytuje editační příkazy pro zásobník příkazů aplikace. Dále vytváří pohled pro zobrazení Chu space modelu procesu.

org.antlr.runtime

Běhové prostředí pro vykonávání pravidel gramatiky nástroje ANTLR. Pro účely validace regulárního výrazu podnikového procesu bylo nutno přidat hlášení chyb při lexikální analýze.

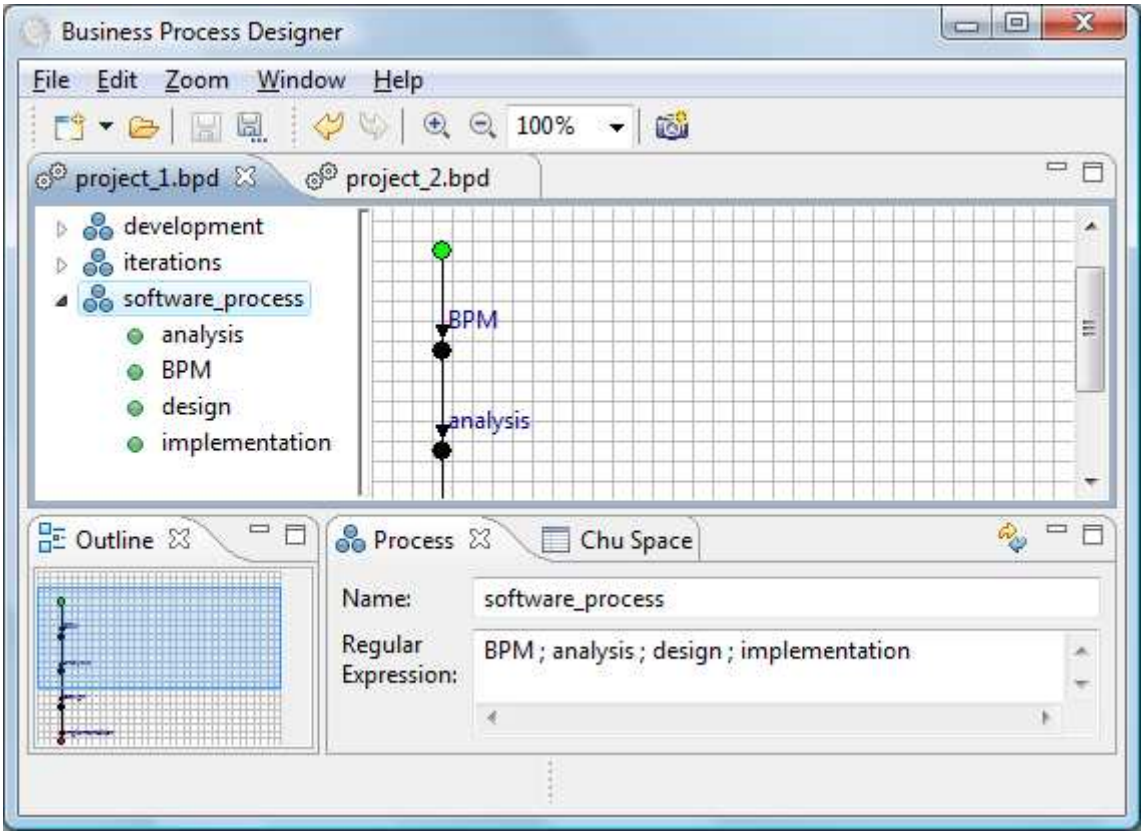
8.3 Hlavní modul BPD

Hlavním modulem BPD je `cz.ds.tools.bpd` a to z toho důvodu, že obsahuje rozšíření `org.eclipse.core.runtime.applications`, které definuje aplikaci jako

takovou. V tomto rozšíření je třeba implementovat rozhraní `org.eclipse.equinox.app.IApplication`, jehož metody `start()` a `stop()` jsou volány při spuštění a ukončení aplikace, což je dáno třídou `cz.ds.tools.bpd.Application`. Dále jsou zde definovány tzv. nastavení produktu pomocí rozšíření `org.eclipse.core.runtime.products`. Jedná se například o jméno aplikace, barvu písma pro texty na úvodním okně spouštění, velikost ukazatele činnosti, ikony okna aplikace atd. Lišta menu a veškeré její položky jsou definovány rozšířením `org.eclipse.ui.menus`. Akcím v položkách menu jsou přiřazeny klávesové zkratky pomocí rozšíření `org.eclipse.ui.bindings`. Uživatelské akce jsou poté definovány v rozšířeních `org.eclipse.ui.actionSets` a `org.eclipse.ui.commands`.

8.4 Rozvržení uživatelského rozhraní

Grafické rozvržení aplikace BPD na obr. 8.5 je dáno pomocí tzv. perspektivy viz kapitola 6.3.



Obr. 8.5: Rozvržení uživatelského rozhraní BPD

Perspektiva BPD obsahuje v horní části okna aplikace oblast editorů pro zobrazování projektových souborů. V dolní části se zleva nachází pohled „Outline“ zobrazující miniaturu grafické části editoru. Napravo od pohledu Outline následuje dvojice pohledů, které jsou umístěny jako záložky a lze se mezi nimi přepínat. První z nich se nazývá „Process“ a obsahuje jméno a regulární výraz aktuálně vybraného podnikového procesu v některém z otevřených editorů. Druhý pohled, pojmenovaný „Chu Space“, zobrazuje Chu space model, který reprezentuje vybraný proces.

Toto grafické rozvržení není konečné. Pozice jednotlivých pohledů anebo otevřených editorů může uživatel měnit uchopením v jeho hlavičce pomocí levého tlačítka myši a přetažením do nové cílové oblasti v okně aplikace.

Definice perspektivy aplikace BPD je dána rozšířením `org.eclipse.ui.perspectiveExtensions` v hlavním modulu aplikace `cz.ds.tools.bpd`.

8.5 Editor projektu

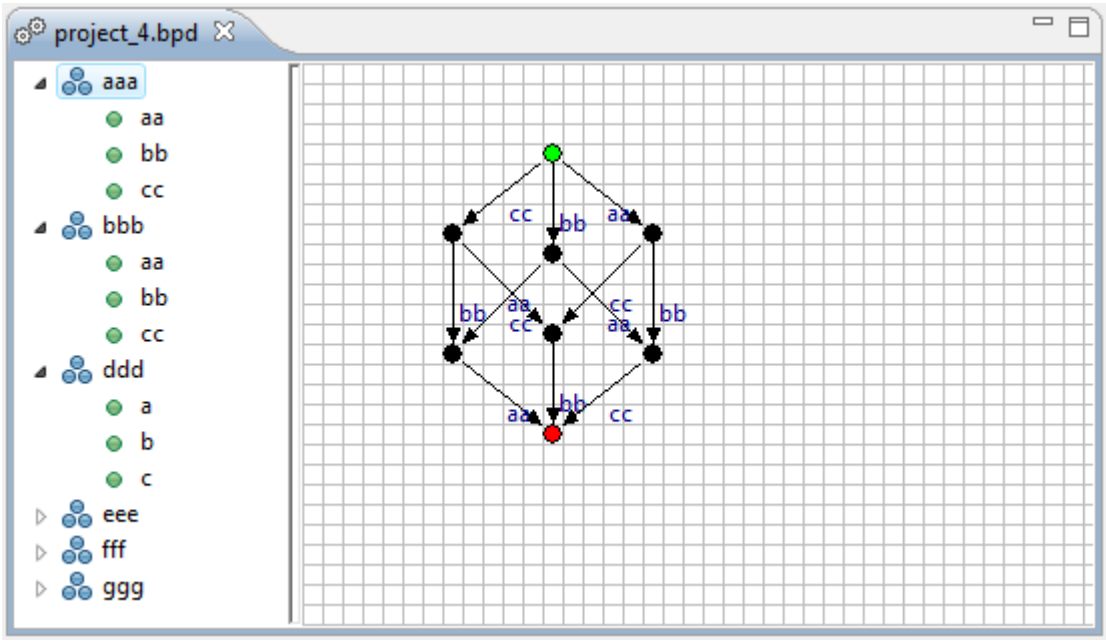
Nejdůležitějším faktorem toho, jak bude editor BPD vypadat a jak se bude chovat, bylo rozhodnutí, zda bude zobrazovat pouze jeden konkrétní podnikový proces (do jednoho souboru by se ukládal právě jeden proces) a správa všech procesů bude tedy podřízena speciálnímu pohledu nad pracovním prostorem v rámci souborového systému aplikace. Anebo se správa procesů přenechá jen editoru. Ten bude zobrazovat tzv. projektový soubor obsahující n procesů, a to jak jejich grafickou reprezentaci, tak i celý seznam procesů načtený z projektu.

První případ je výhodný v tom, že editor bude mít pouze grafickou část, která bude sloužit k zobrazení a editování jen jednoho procesu. O správu všech procesů se postará nezávislý pohled. Avšak toto má svou nevýhodu při vytváření kompozice již existujících procesů. Modely jednotlivých procesů při kompozici by se musely načítat ze souboru. Respektive by tento přístup mohl způsobit kolize v případě, že by některý z procesů otevřený v editoru v neuloženém stavu měl odlišné modely v paměti aplikace oproti souboru. Tudíž by vytvořená kompozice nemusela odpovídat skutečnosti. V neposlední řadě by byla složitější přenositelnost uložených procesů z důvodu nutnosti implementace mechanizmu pro export a import procesů z a do pracovního prostoru aplikace.

Druhý přístup je složitější v tom, že v editoru se bude muset řešit přepínání v zobrazování nebo editování právě vybraného procesu. To však nebude nijak extrémní záležitostí, protože k dispozici bude model všech procesů v projektu. Tato skutečnost též zajistí, že vytváření kompozice z existujících procesů bude vždy založena na aktuálním modelu procesů. Dojde také k zjednodušení přenositelnosti procesů v jednom projektu. Projektový soubor totiž nebude zapotřebí uchovávat v rámci pracovního prostoru aplikace, ale může být načítán z jakéhokoli místa na disku.

Z výše popsaných důvodů jsem zvolil druhý z uvedených přístupů znázorněný na obr. 8.6. Editor se skládá ze dvou hlavních částí. K procházení podnikových procesů je použita

komponenta grafický strom v levé části editoru. Vstupem a tedy i kořenem stromu je modelový objekt projektu aplikace. Zobrazovány jsou ale až jeho uzly v podobě názvů podnikových procesů a jejich listy, což je seznam aktivit daného procesu. V pravé části editoru je pak oblast pro grafické znázornění vybraného procesu ve stromu vlevo. Zde je použita komponenta sloužící k vykreslování grafických prvků, jimiž jsou jednotlivé části vícedimenzionálního automatu zobrazující proces reprezentovány, na plátno.



Obr. 8.6: Editor BPD projektu zobrazující vybraný podnikový proces

Veškerá funkcionalita editoru je implementována v modulu `cz.ds.tools.bpd.editor`. Třída reprezentující jeden otevřený editor se nazývá `cz.ds.tools.bpd.editor.editors.HdaGraphicalEditor`. Ta rozšiřuje abstraktní třídu `org.eclipse.gef.ui.parts.GraphicalEditor`, která vytváří základní kostru grafického editoru. Grafické uživatelské prvky editoru jsou pak vytvářeny v metodě `createPartControl()`. Editor je v aplikaci registrován pomocí rozšíření `org.eclipse.ui.editors`. Zde je vytvářen jak strom k procházení procesů tak i plátno pro vykreslování diagramu procesu.

Samotná reprezentace modelových objektů na plátně je zajištěna pomocí třídy `cz.ds.tools.bpd.editor.part.HdaEditPartFactory`, která implementuje rozhraní `org.eclipse.gef.EditPartFactory`, a to se stará o vytváření grafických prvků pro konkrétní modelové objekty. Všechny třídy, které reprezentují grafické prvky diagramu procesu, se nacházejí ve stejném balíčku `cz.ds.tools.bpd.editor.part` jako výše zmíněná tovární třída.

Akce navázané pouze na editor jsou registrovány ve stejném rozšíření jako editor samotný pod atributem `contributorClass`, který deklaruje implementaci třídy `cz.ds.tools.bpd.editor.actions.HdaEditorActionBarContributor` rozšiřující třídu `org.eclipse.gef.ui.actions.ActionBarContributor`, kde se akce vytvářejí.

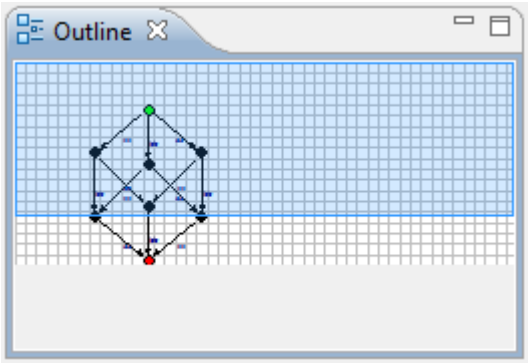
Funkcionalita „zpět“ a „vpřed“ je zajištěna zásobníkem příkazů, který je součástí tzv. editační domény pro daný editor. Ta je v editoru vytvářena metodou `getEditDomain()`. Zásobník příkazů je součástí editační domény a z editoru k němu lze přistoupit pomocí metody `getCommandStack()`. Jednotlivé editační příkazy se na získaném zásobníku příkazů provedou voláním jeho metody `execute()` s parametrem příslušného příkazu, který je potomkem abstraktní třídy `org.eclipse.gef.commands.Command`.

8.6 Pohledy

Jak již bylo zmíněno v kapitole 8.4 aplikace BPD má tři pohledy. Každý z nich zastává specifickou funkci, která bude níže popsána.

8.6.1 Pohled Outline

Pohled Outline zobrazuje miniaturu grafické reprezentace aktuálně vybraného podnikového procesu. Jeho obsah je vytvářen pro každý editor zvlášť. Outline zastává také funkci rychlé navigace po diagramu procesu větších rozměrů, který již nelze zobrazit v okně editoru celý a je pro něj aktivována svislá nebo vodorovná rolovací lišta. Jak je vidět na obr. 8.7 pro rychlou navigaci je vygenerováno modré navigační pole, kterým lze posouvat pomocí stisku levého tlačítka myši.

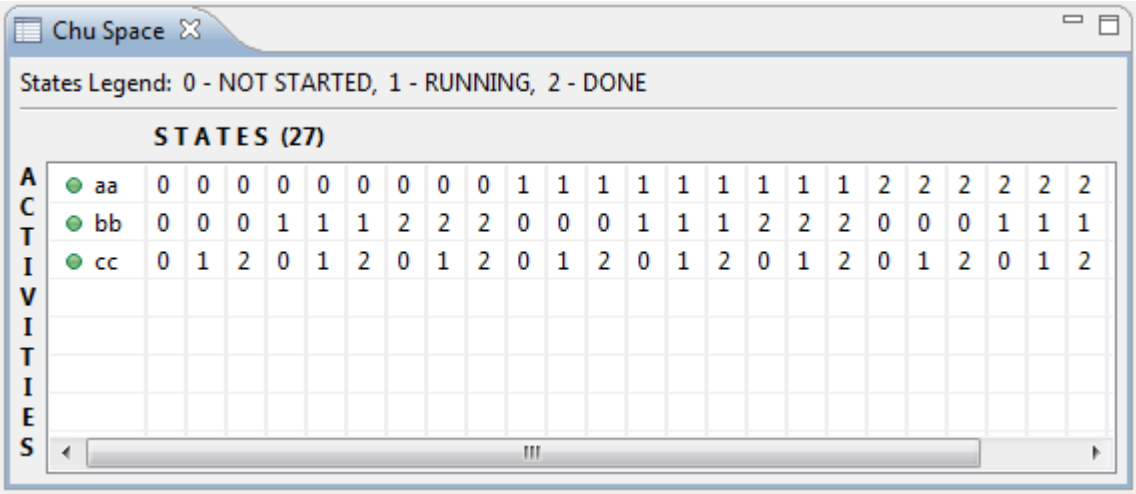


Obr. 8.7: Pohled Outline s modrým navigačním polem

Obsah pohledu Outline je vytvářen automaticky v momentě otevírání grafického editoru. Je reprezentován vnitřní třídou ve třídě editoru nazvanou `OutlinePage`, která rozšiřuje třídu `org.eclipse.gef.ui.parts.ContentOutlinePage`.

8.6.2 Pohled Chu Space

Pohled Chu Space slouží k zobrazení Chu space modelu podnikového procesu (obr. 8.8). K tomuto účelu je použita grafická komponenta tabulka pro zobrazení dat. Jelikož má tabulka čistě informativní charakter, byla z ní odstraněna výchozí funkcionalita pro výběr řádku, sloupce nebo buňky. Obsah pohledu Chu Space je závislý na označeném procesu v některém z otevřených editorů.



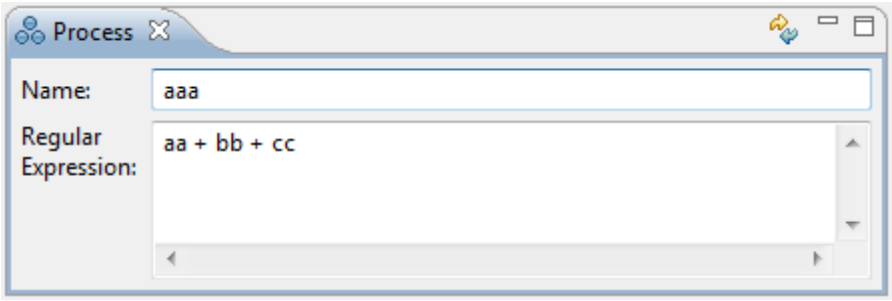
Obr. 8.8: Pohled Chu Space zobrazující Chu space model podnikového procesu

Pohled Chu Space je implementován v modulu `cz.ds.tools.bpd.views` třídou `cz.ds.tools.bpd.views.chuspaceview.ChuSpaceView`, která rozšiřuje abstraktní třídu `org.eclipse.ui.part.PageBookView`. A to z toho důvodu, že poskytuje tzv. více stránkový pohled závislý na aktivní (vybrané) části. V tomto případě na aktivním editoru. Tímto přístupem se zbavujeme nutnosti udržovat kontext, pro který je obsah pohledu zobrazován. Pro jednotlivé stránky pohledu Chu Space jsou pak vytvářeny instance třídy `cz.ds.tools.bpd.views.chuspaceview.ChuSpaceViewPage`, jejíž metoda `createControl()` vytváří grafický obsah pohledu.

8.6.3 Pohled Process

Funkcí pohledu Process je zobrazit jméno a regulární výraz podnikového procesu v editačních polích (obr. 8.9). Skrze tato pole lze tedy měnit v rámci projektu jedinečný název

procesu anebo regulární výraz, který ho definuje. Obě editační pole jsou svázána s příslušnými validačními pravidly, jež ověřují správnost zadaných hodnot přímo při psaní. Princip validace těchto hodnot bude podrobněji popsán v kapitole 8.8. Data zobrazená v pohledu Process jsou stejně jako u pohledu Chu Space závislá na označeném procesu v některém z otevřených editorů.

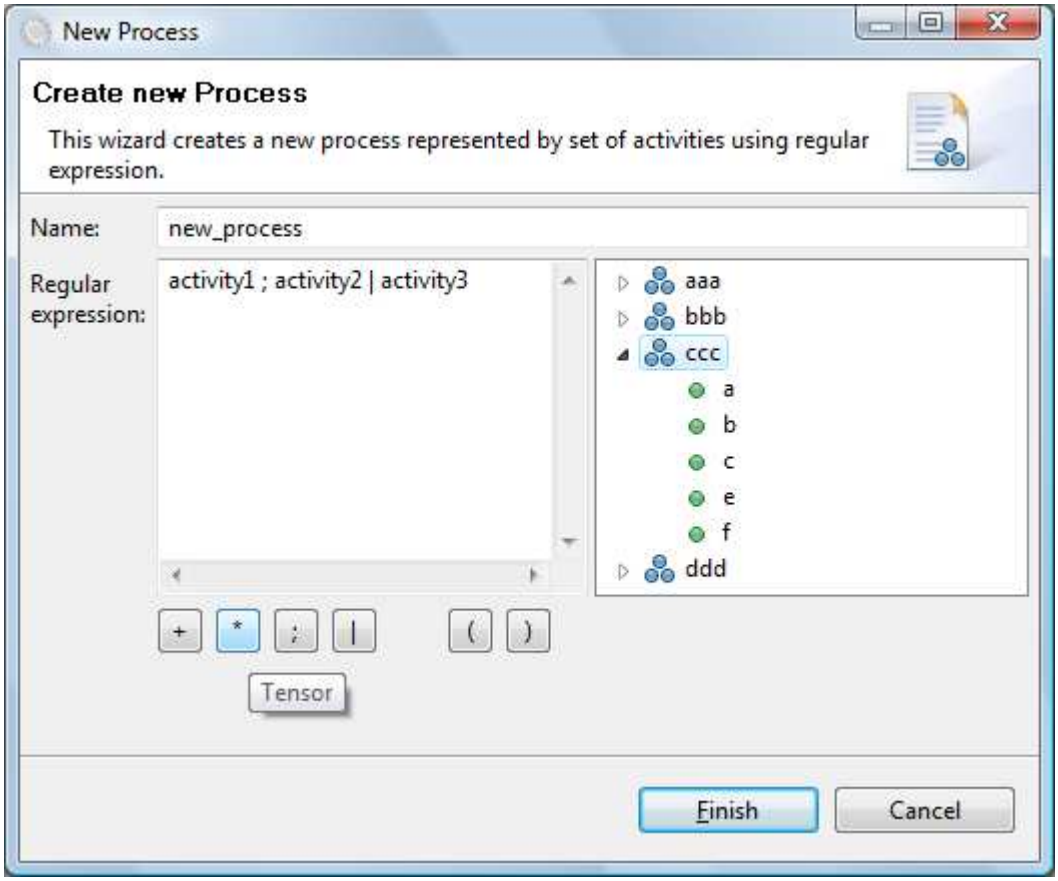


Obr. 8.9: Pohled Process

Pohled Process je implementován v modulu `cz.ds.tools.bpd.views` třídou `cz.ds.tools.bpd.views.processview.ProcessView`, která rozšiřuje abstraktní třídu `org.eclipse.ui.part.PageBookView`. A stejně jako u pohledu Chu Space viz kapitola 8.6.2, jsou jednotlivé stránky tohoto více stránkového pohledu vytvářeny pro každý editor zvlášť třídou `cz.ds.tools.bpd.views.processview.ProcessViewPage`. Tato třída také vytváří editační příkazy popsané v kapitole 8.5 pro změnu názvu nebo regulárního výrazu procesu. Tyto editační příkazy jsou reprezentovány třídami v balíku `cz.ds.tools.bpd.views.processview.commands`.

8.7 Průvodce vytvořením podnikového procesu

Pro vytváření nového podnikového procesu je použita komponenta grafického průvodce. Průvodce obsahuje jednu stránku a ta (mimo hlavičky a navigačních tlačítek) je rozdělena do čtyř částí. Shora první z nich je textové pole pro zadání jména procesu. Pod ním je nalevo situováno další textové pole určené k zadání regulárního výrazu definujícího proces. Obě tato textová pole jsou svázána s validačními pravidly, která budou popsána v kapitole 8.8. Napravo od textového pole regulárního výrazu je pomocí komponenty grafický strom zobrazen seznam procesů a jejich aktivit z projektu aplikace, ve kterém je nový proces právě vytvářen. Tyto již existující procesy mohou být použity ke kompozici nového procesu. A nakonec ve spodní části stránky průvodce se nachází pomocná tlačítka pro vkládání znaků reprezentujících základní operace s Chu space nebo závorky na pozici kurzoru v textovém poli regulárního výrazu. Významy jednotlivých znaků zastupujících operace s Chu space budou vysvětleny v kapitole 8.10. Příklad průvodce vytvořením podnikového procesu je znázorněn na obr. 8.10.



Obr. 8.10: Průvodce vytvořením podnikového procesu

Průvodce vytvořením procesu se nachází v modulu `cz.ds.tools.bpd.editor` a je reprezentován třídou `cz.ds.tools.bpd.create.wizards.NewProcessWizard`, která rozšiřuje abstraktní třídu `org.eclipse.jface.wizard.Wizard`, což je částečná implementace grafického průvodce v Eclipse. Jediná stránka, jak bylo zmíněno výše, je dána třídou `cz.ds.tools.bpd.create.wizards.NewProcessWizerdPage0`.

8.8 Validace vstupních dat

Vstupními daty, která podléhají validaci, jsou v aplikaci BPD název podnikového procesu a regulární výraz definující tento proces. Validace těchto dat je použita při vytváření nového procesu v průvodci vytvořením procesu anebo při jeho editování v pohledu Process. Samotná validace je zajištěna voláním metody `dialogChanged()` ve třídě `cz.ds.tools.bpd.create.wizards.NewProcessWizerdPage0` pro průvodce a ve třídě `cz.ds.tools.bpd.views.processview.ProcessViewPage` pro editování

v pohledu Process. Gramatika k validaci regulárního výrazu se nachází v modulu `cz.ds.tools.bpd.grammar` v souboru `ProcessValidator.g`.

Z důvodu, že běhové prostředí ANTLR nepropouštělo chyby vzniklé při lexikální analýze a to u špatně zadaného jména aktivity, došlo k mírné úpravě zdrojového kódu tohoto běhového prostředí v modulu `org.antlr.runtime`. Jedná se o třídu `org.antlr.runtime.CommonTokenStream`, kde byla přidána proměnná `activityExpMsg` pro uchování aktuálního chybového hlášení výše popsaného problému a metody k nastavení a zjištění hodnoty této proměnné. Daná chyba se odchytává v metodě `fillBuffer()`. Během syntaktické analýzy regulárního výrazu procesu prováděné třídou `cz.ds.tools.bpd.grammar.process.validator.ProcessValidatorParser` může být v metodě `atom()`, která zpracovává jméno aktivity, toto chybové hlášení nastaveno jako výsledek validace.

8.8.1 Validace názvu procesu

Validace názvu procesu kontroluje, zda jsou splněna tato pravidla:

- Název procesu je zadán, jeho hodnota je tedy neprázdná.
- Název procesu je v rámci projektu aplikace jedinečný.
- Název procesu začíná písmenem, a jestliže je delší než jeden znak, pokračuje písmenem, číslicí, tečkou, znakem podtržítko nebo znakem pomlčka, což je dáno následujícím regulárním výrazem $[a-zA-Z][a-zA-Z0-9._-]$*.

8.8.2 Validace regulárního výrazu procesu

Validace regulárního výrazu procesu kontroluje, zda jsou splněna tato pravidla:

- Regulární výraz procesu je zadán, jeho hodnota je tedy neprázdná.
- Regulární výraz procesu odpovídá následující bezkontextové gramatice

  $S \rightarrow T\ C$

  $C \rightarrow D\ T\ C \mid \varepsilon$

  $D \rightarrow * \mid +$

  $T \rightarrow E\ A$

  $A \rightarrow B\ E\ A \mid \varepsilon$

  $B \rightarrow ; \mid '\mid'$

  $E \rightarrow F \mid G \mid '('S')'$

  $F \rightarrow ('a'..'z' \mid 'A'..'Z')\ ('a'..'z' \mid 'A'..'Z' \mid '0'..'9' \mid '.' \mid '-' \mid '\_')$*

  $G \rightarrow \$\ F,$

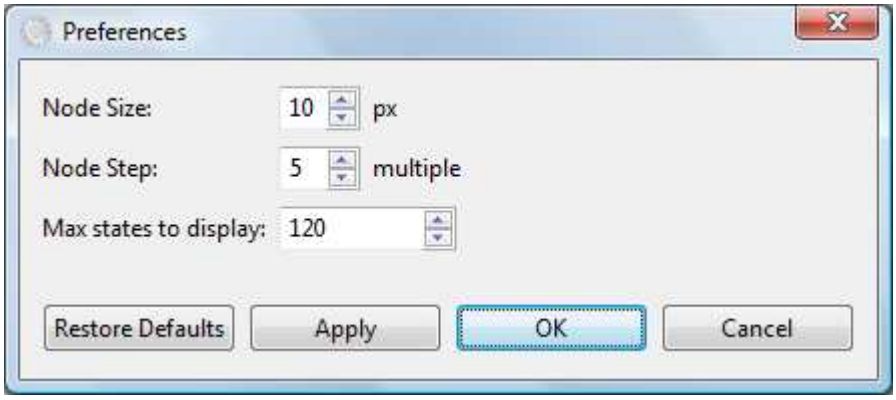
  kde S je startovací symbol a pravidlo F je pro zjednodušení popsáno regulárním výrazem.

8.9 Předvolby a nastavení

BPD implementuje také mechanizmus předvoleb a nastavení, která jsou fyzicky zaznamenávána v souborovém systému aplikace. Jak je vidět na obr. 8.11, uživatel má možnost ovlivnit tato nastavení:

- Rozměr uzlů digramu procesu v bodech obrazovky (pixel).
- Vzdálenost mezi jednotlivými uzly diagramu procesu v násobcích rozměru uzlu. Tato vzdálenost je použita při automatickém generování diagramu přímo z regulárního výrazu procesu.
- Maximální počet stavů vícedimenzionálního automatu zobrazených v tabulce pohledu Chu Space.

První dvě nastavení vlastně ovlivňují výsledek grafického zobrazení automatického generování diagramu procesu. Důvodem posledního nastavení je nadměrné množství stavů v řádech deseti tisíců v procesech s devíti a více paralelními aktivitami.



Obr. 8.11: Předvolby a nastavení v aplikaci BPD

Předvolby jsou implementovány v modulu `cz.ds.tools.bpd.preferences`. Ve třídě `cz.ds.tools.bpd.preferences.ui.PreferencesDialog` se nachází dialogové okno s grafickými prvky pro nastavení jednotlivých hodnot a ve třídě `cz.ds.tools.bpd.preferences.utils.PreferencesUtil` metody pro inicializaci a uložení předvoleb.

8.10 Kompozice procesů v BPD

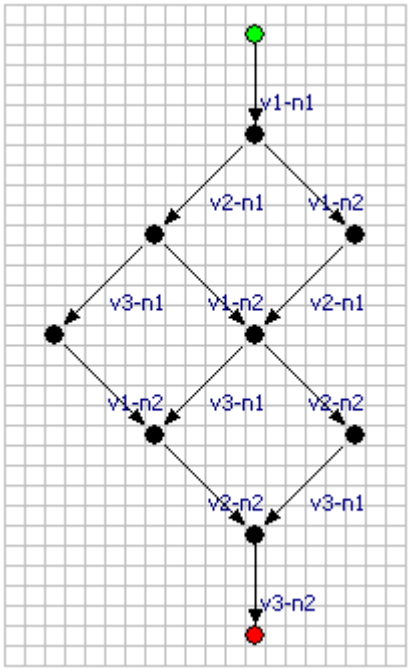
Aplikace BPD podporuje čtyři základní operace procesní algebry realizovatelné pomocí Chu space, které byli popsány v kapitole 3. Každá operace je v regulárním výrazu reprezentována specifickým symbolem (operátorem) mezi základními stavebními bloky (aktivitami) kompozice nebo již existujícími procesy. Při samotném výpočtu kompozice mají operátory tensorového součinu a souběhu vyšší prioritu než operátory zřetězení a výběru, tudíž

mají ve výpočtu přednost. K upřednostnění operací s nižší prioritou před operacemi s prioritou vyšší lze použít kulatých závorek. Jednotlivé operátory představují následující kompozice:

- $*$ – Tensorový součin,
- $+$ – Souběh,
- ; – Zřetězení,
- | – Výběr.

8.10.1 Tensorový součin

Pro ukázku tensorového součinu jsem vybral příklad s vlaky uvedený v kapitole 3.2, který jsem trochu upravil. Po jedné koleji, na které jsou umístěna dvě nádraží, jedou za sebou tři vlaky. Vlak jedoucí za jiným tedy nemůže zastavit na nádraží, kde se ještě nachází vlak předchozí. Definice tohoto příkladu je dána regulárním výrazem $(v1\,;\,v2\,;\,v3)\,*\,(n1\,;\,n2)$, kde $v1$, $v2$ a $v3$ jsou po sobě jedoucí vlaky a $n1$ a $n2$ jsou za sebou ležící nádraží. Výsledek je znázorněn na obr. 8.12.

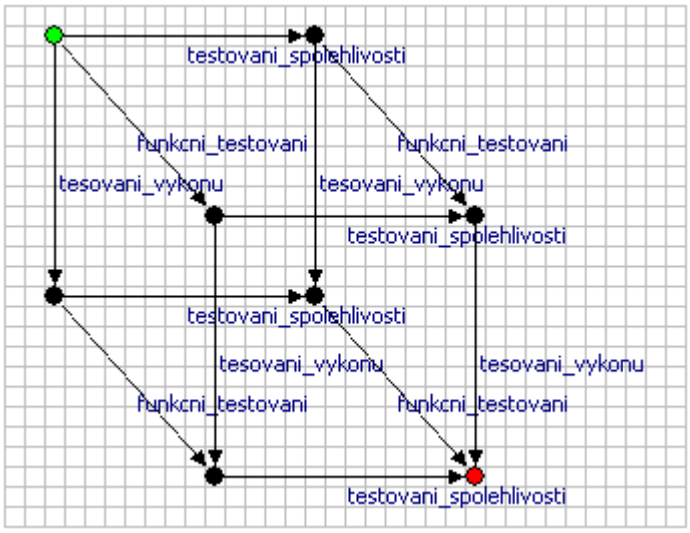


Obr. 8.12: Tensorový součin v BPD

8.10.2 Souběh

Souběh neboli paralelní zpracování lze ukázat na aktivitách, jejichž spuštění je na nich navzájem nezávislé. Mějme proces testování softwarového produktu o třech nezávislých aktivitách funkční testování, testování výkonu a testování spolehlivosti, které musí být všechny

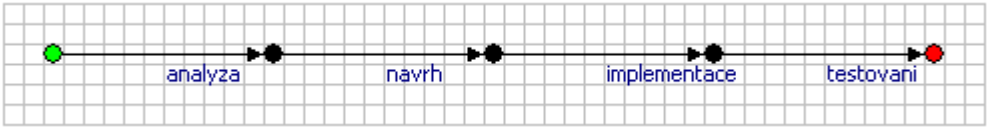
dokončeny, aby proces proběhl celý. Proces definovaný regulárním výrazem $funkcni\_testovani + tesovani\_vykonu + testovani\_spolehlivosti$ je zobrazen na obr. 8.13.



Obr. 8.13: Souběh v BPD

8.10.3 Zřetězení

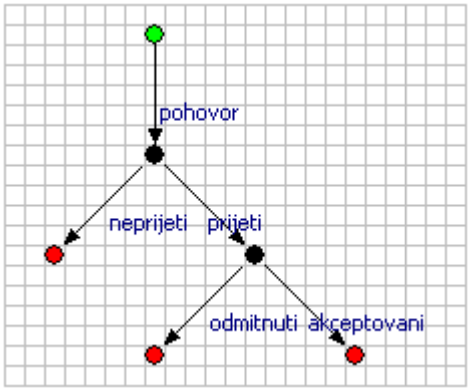
Operace zřetězení zastupující v procesní algebře po sobě jdoucí aktivity nebo procesy, je použita v příkladu procesu vývoje softwaru s aktivitami analýza, návrh, implementace a testování na obr. 8.14. Tento proces odpovídá regulárnímu výrazu $analyza\ ;\ navrh\ ;\ implementace\ ;\ testovani$.



Obr. 8.14: Zřetězení v BPD

8.10.4 Výběr

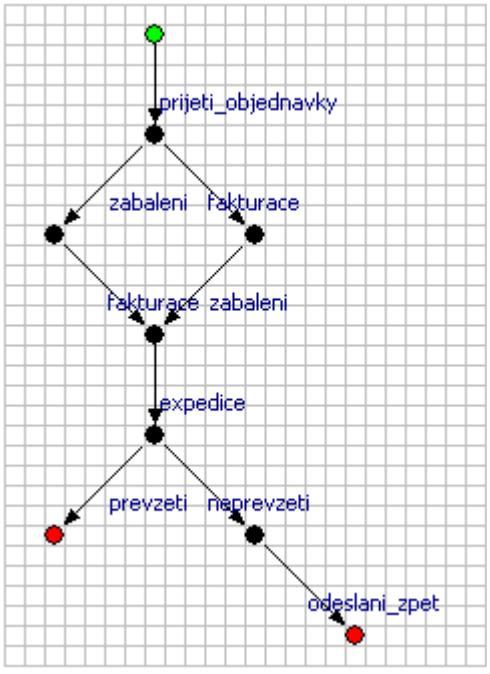
Příkladem operace výběr neboli větvení může být proces přijímacího řízení, jak je tomu na obr. 8.15. Přijímací řízení začíná pohovorem a na jeho základě pokračuje přijetím nebo nepřijetím uchazeče. Ten však ještě v případě jeho přijetí může tuto nabídku akceptovat anebo odmítnout. Všimněme si, že při této operaci, se v procesu zvyšuje počet koncových stavů. Proces je definován regulárním výrazem $pohovor\ ;\ ((prijeti\ ;\ (akceptovani\ |\ odmitnuti))\ |\ neprijeti)$.



Obr. 8.15: Výběr v BPD

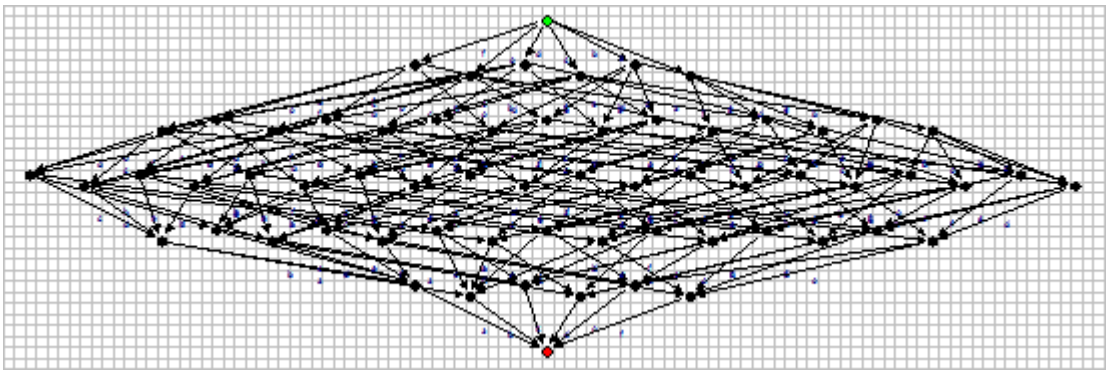
8.10.5 Složitější příklady

Příkladem kombinací operací v kompozici je proces vyřízení objednávky na obr. 8.16. Vyřízení objednávky sestává z přijetí objednávky, po které souběžně následuje zabalení zboží a fakturace. Poté se zboží vyexpeduje na místo doručení, kde si ho adresát buď převezme, nebo nepřevezme. V případě nepřevzetí, je zboží odesláno zpět odesílateli. Proces vyřízení objednávky je dán regulárním výrazem $prijeti\_objednavky\ ;\ fakturace + zabaleni\ ;\ expedice\ ;\ (prevzeti \mid (neprevzeti\ ;\ odeslani\_zpet))$.



Obr. 8.16: Proces vyřízení objednávky v BPD

Dalším příkladem bude proces s šesti fiktivními aktivitami v souběhu znázorněný na obr. 8.17. Proces definuje regulární výraz $a + b + c + d + e + f$. Všimněme si složitosti zobrazeného diagramu. Ta s každou přibývající aktivitou v souběhu roste trojnásobně.



Obr. 8.17: Proces s šesti aktivitami v souběhu

8.11 Gramatika kompozice podnikového procesu

Jak již bylo řečeno v kapitole 6, k vytvoření gramatiky pro kompozice podnikových procesů byl použit nástroj ANTLR. Úkolem této bezkontextové gramatiky je provést sekvenci Chu space operací s operandy, které představují aktivity nebo již existující procesy. Výsledkem výpočtu je nový proces. Jednotlivé operace jsou vykonávány v pořadí zadaným regulárním výrazem s ohledem na prioritu operátorů, jak bylo popsáno v kapitole 8.10. Pravidla gramatiky pro kompozici podnikového procesu v BPD jsou v nástroji ANTLR deklarována následovně:

$S \rightarrow E$

$E \rightarrow F\ (;\ F \mid '|'\ F)^*$

$F \rightarrow A\ (*\ A \mid +\ A)^*$

$A \rightarrow B \mid C \mid (E)$

$B \rightarrow ('a'..'z' \mid 'A'..'Z')\ ('a'..'z' \mid 'A'..'Z' \mid '0'..'9' \mid '.' \mid ' - ' \mid '\_')^*$

$C \rightarrow \$\ B,$

kde S je startovací symbol. Jak je u některých pravidel vidět, ANTLR používá pro jejich deklaraci také regulární výrazy (pravidla E, F, a B). Dále každé pravidlo může vracet nějakou návratovou hodnotu a pro každou větev pravidla může být spouštěn nějaký výkonný kód. Na obr. 8.18 je vidět, jak vypadá deklarace pravidla F v ANTLR.

```
additivesExpr returns [ChuSpace value]:
  e = atom                        { value = e; }
  (
    TENSOR e1 = atom              { value = ChuSpace.tensor(value, e1); }
    | ADDITIVES e1 = atom         { value = ChuSpace.additives(value, e1); }
  )*
;
```

Obr. 8.18: Ukázka deklarace pravidla gramatiky v ANTLR

Gramatika pro kompozici podnikového procesu se nachází v modulu `cz.ds.tools.bpd.grammar` v souboru `Process.g`. K samotnému vytvoření Chu space z regulárního výrazu procesu slouží metoda `createChuSpace()` ve třídě `cz.ds.tools.bpd.grammar.process.ProcessParserFactory`. Výkonné metody pro jednotlivé operace s Chu space a zároveň datový model k uchovávání mezivýsledku se nacházejí ve třídě `cz.ds.tools.bpd.chuspace.ChuSpace` v modulu `cz.ds.tools.bpd.chuspace`. Pro vytvoření grafické reprezentace procesu z Chu space slouží třída `cz.ds.tools.bpd.chuspace.hda.HdaGeneratorFactory` a její metoda `createHda()`.

8.12 Nápověda

Nápověda je implementována v modulu `cz.ds.tools.bpd.help` pomocí rozšíření `org.eclipse.help.toc`. Eclipse poskytuje vlastní mechanizmus zobrazení nápovědy v samostatném okně. Tento mechanizmus je založený na prezentaci webového prohlížeče a proto má plnou podporu HTML stránek. V okně nápovědy je automaticky zahrnuto vyhledávání, kontextová nápověda, indexování klíčových slov a navigační systém. Každá stránka nápovědy je tedy tvořena HTML stránkou a mezi jednotlivými stránkami se lze přepínat pomocí odkazů obsahu nápovědy nebo odkazů na stránkách.

9 Závěr

Je důležité uvědomit si, čeho chce člověk dosáhnout. Mým cílem bylo vytvořit funkční a přehlednou aplikaci a obohatit čtenáře o zajímavé poznatky ať už z oblasti modelování podnikových procesů nebo vývoje samostatných aplikací s bohatým grafickým rozhraním.

V práci je popsána možnost modelovat podnikové procesy pomocí vícedimenzionálních automatů. K reprezentaci modelu vícedimenzionálního automatu je využito matematické teorie Chu space a jsou definovány základní operace procesní algebry nad těmito Chu space.

Použití Chu space umožňuje držení modelu podnikového procesu v jednoduchých datových strukturách jako matici. Díky tomu jsou i složité kompozice procesů realizovány ve velice krátkém čase. Faktorem, který na složitost působí nejvíce, je souběh aktivit nebo procesů. Při souběhu do deseti aktivit včetně, je výsledná matice vypočtena stále v reálném čase. Problém nastává při překročení této hodnoty, kdy počet sloupců matice začíná dosahovat stotisícových řádů a s každou další paralelní aktivitou roste trojnásobně. To má samozřejmě kritický dopad pro paměť použitou k výpočtu, která nemusí být dostačující. Otázkou je, zda se maximální počet deseti souběžných aktivit v jednom procesu jeví jako dostačující pro výpočty probíhající v reálném čase. Nebo naopak kolik procesů v praxi obsahuje více než deset souběžných aktivit.

Dalším výkonnostním omezením je generování vícedimenzionálního automatu z Chu space. To začíná být náročnější na čas již při počtu stavů automatu v řádech desetitisíců.

I když je výsledný vícedimenzionální automat procesu přehledný a srozumitelný, stále zůstává nevyřešeným problémem vizualizace více jak tří souběžných aktivit.

Během implementace se potvrdilo, že platforma Eclipse nabízí v dnešní době silnou podporu pro vývoj aplikací s bohatým grafickým rozhraním počínaje definicí modelů aplikace a konče sestavením samotného produktu. Navíc poskytuje modulární systém skladby aplikace, což umožňuje její snadnější rozšiřitelnost.

10 Literatura

1. **Ježek, David.** Použití vícedimenzionálních automatů k dolování podnikových procesů. *Dizertační práce.* Ostrava : VŠB - Technická Univerzita Ostrava. Vedoucí dizertační práce prof. Ing. Ivo Vondrák CsC, 2008.

2. *Modeling Concurrency with Geometry.* **Pratt, Vaughan R.** Orlando, Florida, 1991, Conference Record of the Eighteenth Annual ACM Symposium on Principles of Programming Languages, stránky 311 - 322.

3. **Goubault, Eric a Jensen, Thomas P.** Homology of Higher Dimensional Automata. *International Conference on Concurrency Theory.* 1992. stránky 254-268.

4. **van Glabbeek, Rob.** *Bisimulation semantics for higher dimensional automata. Technical report.*, Stanford University, 1991.

5. **Pratt, Vaughan R.** Chu spaces. *Course notes for the School in Category Theory and Applications.* Coimbra, 1999.

6. **Girard, J.-Y.** Linear logic. *Theoretical Computer Science.* 1987, Sv. 50, 1-102.

7. **Gupta, Vineet a Pratt, Vaughan R.** Gates accept concurrent behavior. *Proc. 34th Ann. IEEE Symp. on Foundations of Comp. Sci.* 1993, 62-71.

8. **Gupta, Vineet.** Chu Spaces: A Model of Concurrency. *Dizertační práce.* Stanford University, 1994. CS-TR-94-1521.

9. **Pratt, Vaughan R.** Modeling Concurrency with Partial Orders. *International Journal of Parallel Programming.* 1986, Sv. 15, stránky 33 - 71.

10. **Pratt, Vaughan R.** Higher dimensional automata revisited. *Math. Structures in Comp. Sci.* 2000. Sv. 10, stránky 525-548.

11. **Frej, Dieter a Parr, Terence.** ANTLR v3 documentation. *ANTLR v3 documentation - ANTLR 3 - ANTLR Project.* [Online] 23. Září 2009. [Citace: 9. Duben 2010.] http://www.antlr.org/wiki/display/ANTLR3/ANTLR+v3+documentation.

12. **McAffer, Jeff a Lemieux, Jean-Michel.** *Eclipse Rich Client Platform: Designing, Coding, and Packaging Java™ Applications.* Addison-Wesley Professional, 2005. ISBN 0-321-33461-2.

13. **Clayberg, Eric a Rubel, Dan.** *Eclipse Plug-ins, Third Edition.* Addison-Wesley Professional, 2008. ISBN 0-321-55346-2.

14. **Steinberg, Dave, a další.** *EMF: Eclipse Modeling Framework, Second Edition.* Addison-Wesley Professional, 2008. ISBN 0-321-33188-5.

15. **Aniszczyk, Chris a Hudson, Randy.** Create an Eclipse-based application using the Graphical Editing Framework. [Online] IBM, 27. Květen 2007. [Citace: 11. Duben 2010.] http://www.ibm.com/developerworks/library/os-eclipse-gef11/.

16. About The ANTLR Parser Generator. [Online] [Citace: 14. Duben 2010.] http://www.antlr.org/about.html.

17. Eclipse Downloads. [Online] The Eclipse Foundation. [Citace: 12. Duben 2010.] http://www.eclipse.org/downloads/.

18. Java Glossary : Adapter Classes. [Online] RoseIndia. [Citace: 14. Duben 2010.] http://www.roseindia.net/help/java/a/adapter-classes.shtml.

19. XML Metadata Interchange. *Wikipedia, the free encyclopedia.* [Online] 8. Listopad 2002. [Citace: 11. Duben 2010.] http://en.wikipedia.org/wiki/XML_Metadata_Interchange.

A Uživatelská příručka

Business Process Designer je aplikace pro vytváření modelů podnikových procesů. Procesy jsou spravovány pomocí projektů, které je možno uložit a opětovně načítat. Modely procesů jsou vytvářeny na základě kompozice základních stavebních bloků (aktivit) nebo již existujících procesů. K samotnému výpočtu kompozice se využívá matematického modelu Chu space. Definice kompozice se provádí pomocí regulárního výrazu, kde jednotlivé operátory představují operace procesní algebry. Výsledný model kompozice je graficky zobrazen jako vícedimenzionální automat. Grafické zobrazení lze dále upravovat za účelem dosažení přehlednější vizualizace procesu.

Seznam základních funkcí aplikace je následující:

- vytvoření projektu spravujícího podnikové procesy,
- vytváření podnikových procesů v projektu,
- grafické zobrazení vytvořeného podnikového procesu,
- editování podnikového procesu,
    - editování výrazu definující proces,
    - editování grafického zobrazení procesu,
- smazání podnikového procesu,
- uložení projektu,
- otevření (načtení) projektu,
- uložení grafického zobrazení procesu jako obrázek,
- zobrazení Chu space modelu procesu,
- další pomocné grafické a editační funkce,
    - zobrazení miniatury grafického zobrazení procesu,
    - funkce „zoom“,
    - funkcionalita „zpět/znovu“.

A.1 Spuštění a ukončení aplikace

Aplikace se spouští souborem `bpd.exe` v jejím kořenovém adresáři. Ukončit aplikaci lze výběrem položky menu **File > Exit**, standardním tlačítkem pro ukončení aplikace v pravém horním rohu na liště hlavního okna nebo klávesovou zkratkou **Ctrl+Q**.

A.2 Projekt BPD

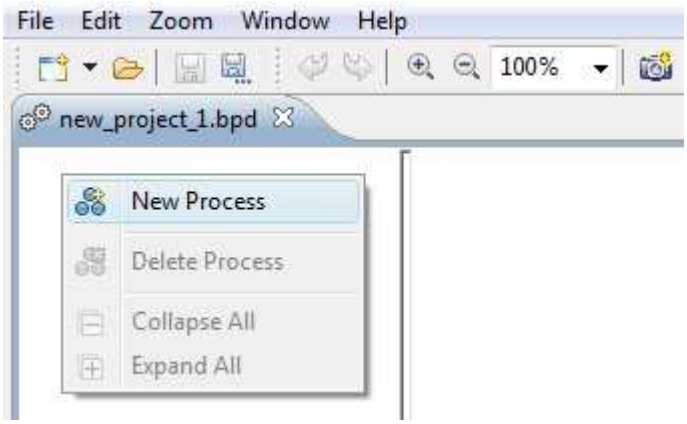
Nový projekt – výběrem položky menu **File > New > BPD Project**. Nebo na nástrojové liště výběrem ikony  a zvolením položky  **BPD Project**. Nový projekt má výchozí název a je v neuloženém stavu. Po prvním uložení se jeho název nastaví dle jména souboru.

Uložení projektu – výběrem položky menu **File > Save**, na nástrojové liště výběrem ikony nebo klávesovou zkratkou **Ctrl+S**. Případně pro „uložení jako“ výběrem položky menu **File > Save As…** nebo na nástrojové liště výběrem ikony .

Otevření projektu – výběrem položky menu **File > Open…** nebo na nástrojové liště výběrem ikony . V následně otevřeném dialogovém okně pro výběr souboru zvolením požadovaného a potvrzením příslušným tlačítkem dialogu.

A.3 Podnikový proces

Nový proces – pro otevřený projekt výběrem menu **File > New > Process** nebo na nástrojové liště výběrem ikony a zvolením položky **Process**. Nebo přímo v levé části editoru otevřeného projektu výběrem položky **New Process** v kontextovém menu (viz Obr. A.1). Následně se otevře průvodce vytvořením nového procesu (viz Obr. A.2).



Obr. A.1: Nový proces pomocí kontextového menu editoru

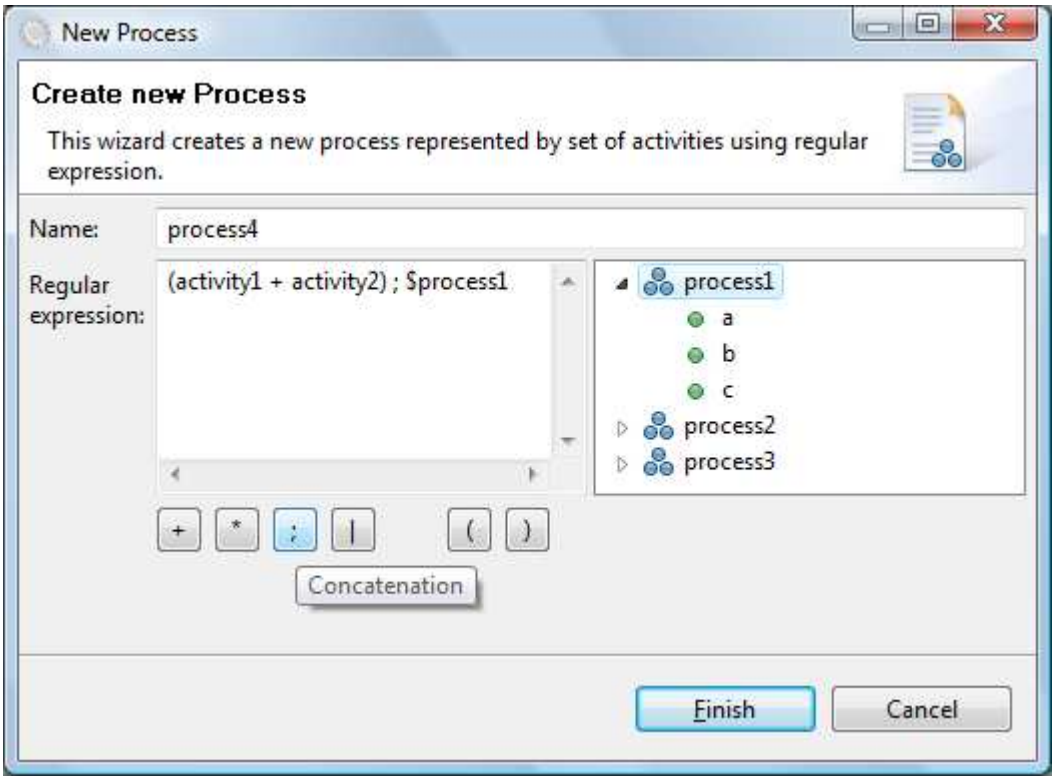
Průvodce novým procesem – v průvodci, který má jednu stránku (Obr. A.2), je nutné vyplnit pole **Name**, kde se zadává název vytvářeného procesu. Název procesu musí začínat písmenem (nerozlišují se malá a velká písmena) a je-li delší než jeden znak může pokračovat písmenem, číslicí nebo znaky ‘.’, ‘-’ a ‘_’. Dále musí být zadáno pole **Regular expression** a to regulárním výrazem, který definuje podnikový proces. Regulární výraz sestává z názvů aktivit a referencí na již existující procesy oddělených operátory reprezentující operace procesní algebry. Části kompozice mohou být uzavřeny v kulatých závorkách k určení priority při samotném výpočtu. Jednotlivé operátory představují následující kompozice:

- `*` – Tensorový součin,
- `+` – Souběh,
- `;` – Zřetězení,
- `|` – Výběr.

Operátory tensorového součinu a souběhu mají vyšší prioritu než operátory zřetězení a výběru, tudíž jsou ve výpočtu upřednostňovány i bez použití závorek. Názvy aktivit použité v regulárním výrazu podléhají stejným pravidlům jako názvy procesů popsané výše. Taktéž reference na již existující proces s tím, že před název takového procesu je umístěn znak '$'.

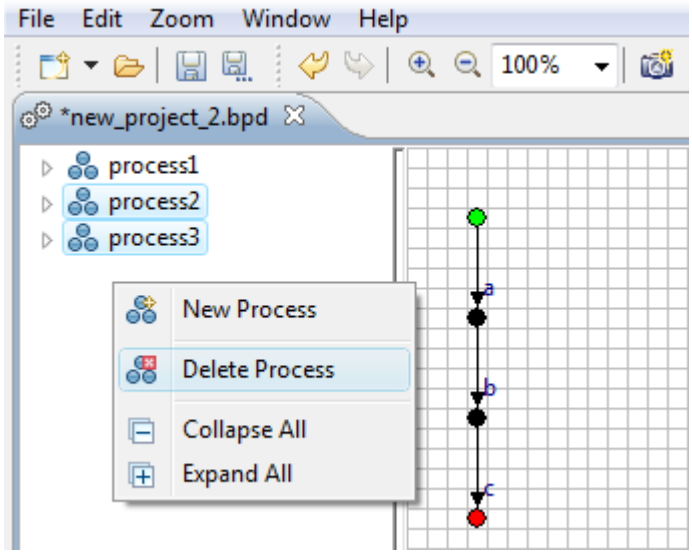
K usnadnění tvorby regulárního výrazu jsou pod polem pro jeho zadání umístěna tlačítka dostupných operací a závorek. Stiskem některého z tlačítek se vloží příslušný znak operace nebo závorky v místě poslední pozice kurzoru. Další možností je využití stromu existujících procesů napravo od pole pro regulární výraz. Pomocí dvojkliku na některý z procesů se vloží jeho reference opět v místě poslední pozice kurzoru pole regulárního výrazu. Stejným způsobem se vloží název aktivity existujícího procesu po dvojkliku na ni.

Jsou-li obě potřebná pole správně vyplněna, je zpřístupněno tlačítko **Finish** ve spodní části průvodce, po jehož stisknutí se spustí výpočet kompozice.



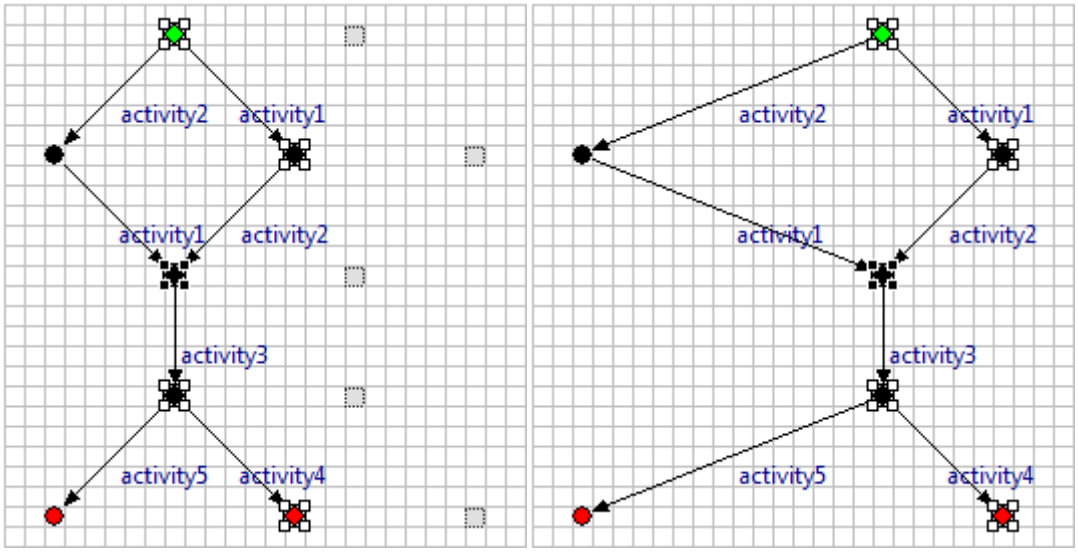
Obr. A.2: Průvodce novým procesem

Odstranění procesu – v otevřeném projektu výběrem položky **Delete Process** z kontextového menu v levé části editoru pro jeden a více označených procesů (viz Obr. A.3).



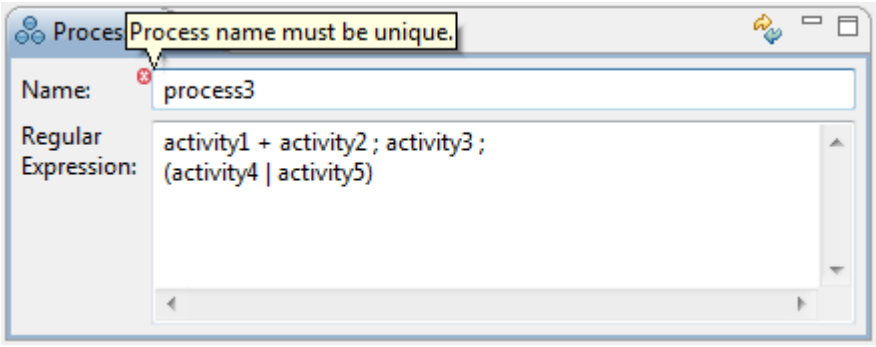
Obr. A.3: Odstranění procesu pomocí kontextového menu editoru

Editování procesu – rozmístění grafického zobrazení procesu lze měnit uchopením jednoho nebo více počátečních či koncových stavů aktivit a jejich přetažením na jinou pozici v pravé zobrazovací části editoru, jak je vidět na Obr. A.4.



Obr. A.4: Editování grafického zobrazení procesu

Název a regulární výraz procesu lze upravovat v pohledu Process (viz Obr. A.5). Pro zaregistrování změny systémem je nutné po úpravě daného editačního pole potvrdit editační příkaz klávesou Enter anebo se přepnout do jakéhokoli jiného pole či okna aplikace.

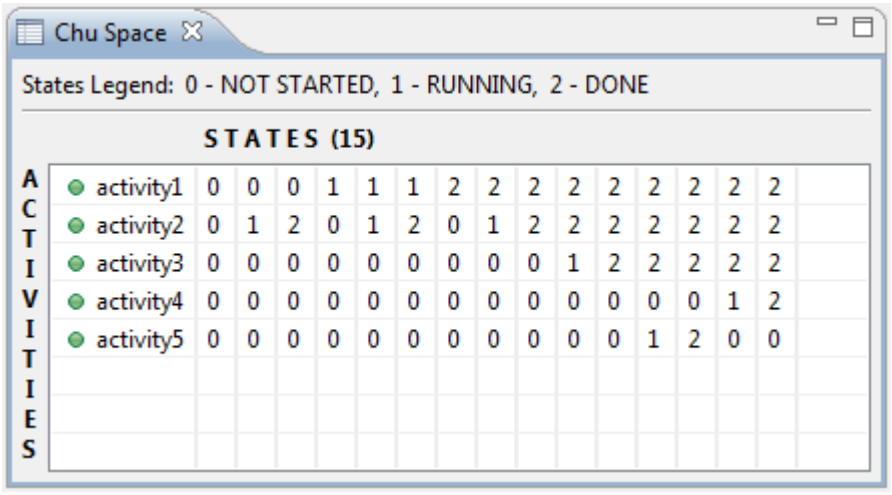


Obr. A.5: Pohled Process s chybovým hlášením

Grafické zobrazení procesu lze kdykoli vykreslit do výchozího rozložení tak, jak ho generuje systém při jeho vytváření. K tomu slouží tlačítko **Refresh layout** na liště pohledu Process.

A.4 Model Chu space

K zobrazení Chu space modelu podnikového procesu slouží pohled Chu Space. Tento pohled obsahuje tabulku, kde každý sloupec představuje jeden stav procesu a řádky stavy aktivit pro každý stav procesu (viz Obr. A.6).



Obr. A.6: Pohled Chu Space

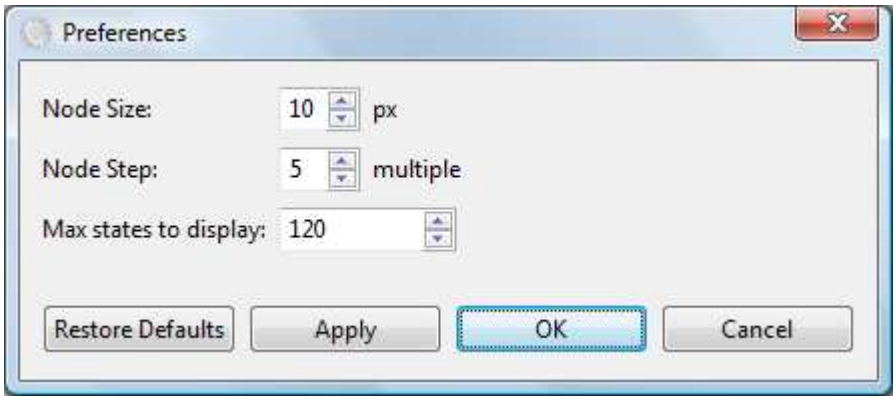
A.5 Export procesu

Vytvořený podnikový proces lze exportovat jako obrázek ve formátech BMP, JPEG a PNG. Právě zobrazený proces je exportován výběrem menu **File > Save As Image** nebo na

nástrojové liště výběrem ikony . V následně otevřeném dialogovém okně pro výběr cílové složky a názvu souboru je možné zvolit požadovaný formát obrázku a potvrzením příslušným tlačítkem dialogu se proces exportuje.

A.6 Předvolby a nastavení

Aplikace má několik předvoleb, které lze změnit v příslušném dialogovém okně (Obr. A.7) v menu **Window > Preferences**. V poli **Node Size** rozměr uzlů digramu procesu v bodech obrazovky (pixel). V poli **Node Step** vzdálenost mezi jednotlivými uzly diagramu procesu v násobcích rozměru uzlu. Tato vzdálenost je použita při automatickém generování diagramu přímo z regulárního výrazu procesu. V poli **Max states to display** maximální počet stavů vícedimenzionálního automatu zobrazených v tabulce pohledu Chu Space.



Obr. A.7: Dialogové okno předvoleb

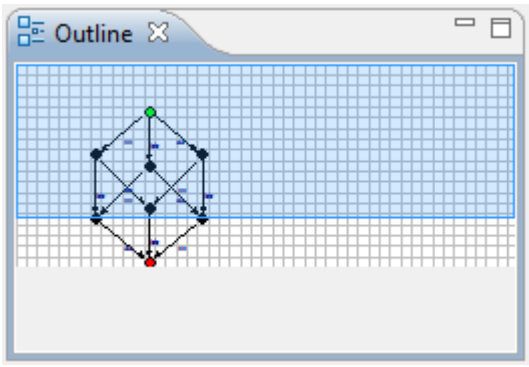
A.7 Funkce zpět/znovu

Příkazy „zpět“ a „znovu“ jsou dostupné v menu **Edit > Undo** a **Edit > Redo**, na nástrojové liště ikony a nebo pomocí klávesových zkratek **Ctrl+Z** a **Ctrl+Y**. Na tyto příkazy jsou vázány následující editační příkazy:

- vytvoření nového procesu,
- smazání procesu,
- editování grafického zobrazení procesu,
- editování názvu procesu,
- editování regulárního výrazu procesu,
- vykreslení výchozího rozložení grafického zobrazení.

A.8 Navigace a zoom

K rychlé navigaci slouží pohled Outline, který zobrazuje miniaturu diagramu právě vybraného procesu. Pro diagram větších rozměrů, který již nelze zobrazit v okně editoru celý, je v pohledu Outline vygenerováno modré navigační pole, kterým lze posouvat pomocí stisku levého tlačítka myši a měnit tak zobrazenou část diagramu v editoru (Obr. A.8).



Obr. A.8: Navigační pole v pohledu Outline

Pro přiblížení nebo oddálení grafického zobrazení procesu slouží funkcionalita zoom. Ta je dostupná v menu **Zoom > Zoom In** a **Zoom > Zoom Out**, na nástrojové liště ikony a nebo pomocí klávesových zkratek **Ctrl++** a **Ctrl+–**. Na nástrojové liště vedle tlačítek přiblížení a oddálení je navíc rozbalovací nabídka s přednastavenými hodnotami pro zoom.

A.9 Perspektiva

Perspektiva BPD definuje uspořádání pohledů aplikace. Pomocí uchopení za hlavičku pohledu a přetažením na jinou pozici lze toto rozvržení měnit. Reset pohledů do výchozího uspořádání se provádí výběrem menu **Window > Reset Perspective…**.

Každý pohled lze zavřít tlačítkem v jeho hlavičce. Zobrazení pohledu je možné výběrem požadovaného v menu **Window > Show View**.

B Obsah přiloženého CD

bpd – aplikace Business Process Designer
projects – ukázkové projekty aplikace BPD
src – zdrojové soubory aplikace BPD jako Eclipse projekty
doc – programátorská dokumentace ve formátu „javadoc“
zadani.pdf – zadání diplomové práce
diplomova_prace.docx – text diplomové práce
diplomova_prace.pdf – text diplomové práce